



freeBSDTM JOURNAL

July / August 2015

FreeBSD in the **ENTERPRISE**

- Groupon's Deal on FreeBSD
- The Isilon Experience

ALSO

Reflections on
FreeBSD.org
Packages



Security Gateways Without Limits



SG-4860 1U

Value 1U Security Gateway for small and branch offices
Connect VPN tunnels between offices or with Amazon EC2®
4 Core Intel® C2558 CPU with AES-NI support, Standard 8GB memory, 4x 1GbE Ethernet Ports, expandable storage
From \$799



SG-8860 1U

Low power 1U Security Gateway for businesses with high network loads
Multi-WAN, no artificial limits, optional IDS/IPS packages available
8 Core Intel® C2758 CPU with AES-NI support
Standard 8GB memory, 6x 1GbE Ethernet Ports, 64GB disk included
From \$999



C2758

Sturdy 1U gateway for businesses with expansion options
Scales from firewall to UTM with flexible software options
8 Core Intel® C2758 CPU with AES-NI support, 4x 1GbE Ethernet Ports
Expandable memory, storage and 1 Gigabit or 10 Gigabit Ethernet cards
From \$1399



XG-1540 1U

Top of the line 1U Security Gateway for complex networks
8 Core Intel® Xeon® D-1540 processor with AES-NI support, 4x 16 GB memory and 120GB Intel SSD
Dual 1GbE Ethernet Ports, Dual 10GbBaseT Ports built-in
Expandable memory, storage and networking options
From \$2499

pfSense scales with your business at no additional cost providing unlimited firewall rules, unlimited users, unlimited VPN connections.
Why pay more for limited security?

Shop now at the official pfSense store or authorized partners worldwide.

7212 McNeil Drive Suite 204 Austin, TX 78729 +1.512.646.4100

pfSense® is a registered trademark of Electric Sheep Fencing, LLC.

Amazon EC2 and Amazon are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries.



Table of Contents

FreeBSD Journal

Vol. 2, Issue No. 4

July/August 2015

FreeBSD in the

ENTERPRISE

Columns & Departments

3 Foundation Letter FreeBSD in the Enterprise.
By George Neville-Neil

30 Conference Report BSDCan 2015.
Stephen Bourne's talk about the creation of the Bourne shell and his many insights and inside stories was the perfect beginning to an incredible conference.
By Shonali Balakrishna

33 Ports Report Activity on ports was very high during the May-June period, with an increase of more than 20% over the March-April period! There were also a lot of improvements to the ports framework. By Frederic Culot

34 Book Review A Review of *The Practice of System and Network Administration* by Thomas A. Limoncelli, Christina J. Hogan and Strata R. Chalup.
By Greg Lehey

36 svn update Thanks to the FreeBSD community and FreeBSD developers for the hard work that went into the FreeBSD 10.2-RELEASE. By Glen Barber

38 This Month in FreeBSD
The author participated in a FreeBSD Hackathon at the Linuxhotel in Essen, Germany, and interviewed Lars Engels (lme@), one of the organizers, to get his thoughts about the event. By Dru Lavigne

40 Events Calendar By Dru Lavigne

**Interacting
with the FreeBSD Project**

26 Inside the FreeBSD Foundation

We are determined to make more people aware of FreeBSD and encourage people to try it. By Deb Goodkin

4 Groupon's Deal on FreeBSD
Introducing technical changes requires successfully navigating the necessary organizational change elements. By Sean Chittenden

14 FreeBSD: The Isilon EXPERIENCE
Each of those nodes that Isilon sells, and that they've been selling for 10 years or so now, runs FreeBSD.
By Benno Rice

20 Reflections on FreeBSD.org PACKAGES

Since the author's first article on packages, he has faced many challenges, made many changes, learned some lessons, and imagined a future of doing things differently. By Bryan Drewery

FreeNAS

in an Enterprise Environment

By the time you're reading this, FreeNAS has been downloaded more than 5.5 million times. For home users, it's become an indispensable part of their daily lives, akin to the DVR. Meanwhile, all over the world, thousands of businesses, universities, and government departments use FreeNAS to build effective storage solutions in myriad applications.



What you will learn...

- How TrueNAS builds off the strong points of the FreeBSD and FreeNAS operating systems
- How TrueNAS meets modern storage challenges for enterprise

The FreeNAS operating system is free to the public and offers thorough documentation, an active community, and a feature-rich storage environment. Based on FreeBSD, it can share over a host of protocols (SMB, FTP, iSCSI, etc) and features an intuitive interface. The ZFS file system, a plug-in system for much more.

Despite the massive popularity of FreeNAS, many aren't aware of its big brother dutifully protecting data in some of the most demanding environments: the proven, enterprise-grade, professionally-supported line of TrueNAS.

But what makes TrueNAS different? Well, I'm glad you asked...

Commercial Grade Support

When a mission critical storage system fails, an organization's whole operation can halt. Whole community-based (free), it can't always get an expert and running in a timely manner. Responsiveness and expert support are dedicated support team to provide that safety.

Created by the same team that developed FreeNAS.

WE INTERRUPT THIS MAGAZINE TO BRING YOU THIS IMPORTANT ANNOUNCEMENT:

THE PEOPLE WHO DEVELOP FREENAS, THE WORLD'S MOST POPULAR STORAGE OS, HAVE JUST REVAMPED TRUENAS.



POWER WITHOUT CONTROL MEANS NOTHING. TRUENAS STORAGE GIVES YOU BOTH.

- | | |
|---|--|
| <input checked="" type="checkbox"/> Simple Management | <input checked="" type="checkbox"/> Self-Healing Filesystem |
| <input checked="" type="checkbox"/> Hybrid Flash Acceleration | <input checked="" type="checkbox"/> High Availability |
| <input checked="" type="checkbox"/> Intelligent Compression | <input checked="" type="checkbox"/> Qualified for VMware and HyperV |
| <input checked="" type="checkbox"/> All Features Provided Up Front (no hidden licensing fees) | <input checked="" type="checkbox"/> Works Great With Citrix XenServer® |

To learn more, visit: www.ixsystems.com/truenas



POWERED BY INTEL® XEON® PROCESSORS

Intel, the Intel logo, Intel Xeon and Intel Xeon Inside are trademarks of Intel Corporation in the U.S. and/or other countries.

VMware and VMware Ready are registered trademarks or trademarks of VMware, Inc. in the United States and other jurisdictions.

Citrix makes and you receive no representations or warranties of any kind with respect to the third party products, its functionality, the test(s) or the results therefrom, whether expressed, implied, statutory or otherwise, including without limitation those of fitness for a particular purpose, merchantability, non-infringement or title. To the extent permitted by applicable law, in no event shall Citrix be liable for any damages of any kind whatsoever arising out of your use of the third party product, whether direct, indirect, special, consequential, incidental, multiple, punitive or other damages.

FreeBSDTM JOURNAL

Editorial Board



- John Baldwin • Member of the FreeBSD Core Team
- Justin Gibbs • Founder and President of the FreeBSD Foundation and a senior software architect at Spectra Logic Corporation
- Daichi Goto • Director at BSD Consulting Inc. (Tokyo)
- Joseph Kong • Author of *FreeBSD Device Drivers*
- Dru Lavigne • Director of the FreeBSD Foundation and Chair of the BSD Certification Group
- Michael W. Lucas • Author of *Absolute FreeBSD*
- Kirk McKusick • Director of the FreeBSD Foundation and lead author of *The Design and Implementation* book series
- George Neville-Neil • Director of the FreeBSD Foundation and co-author of *The Design and Implementation of the FreeBSD Operating System*
- Hiroki Sato • Director of the FreeBSD Foundation, Chair of AsiaBSDCon, member of the FreeBSD Core Team and Assistant Professor at Tokyo Institute of Technology
- Robert Watson • Director of the FreeBSD Foundation, Founder of the TrustedBSD Project and Lecturer at the University of Cambridge

S&W PUBLISHING LLC
PO BOX 408, BELFAST, MAINE 04915

- Publisher** • Walter Andrzejewski
walter@freebsdjournal.com
- Editor-at-Large** • James Maurer
jmaurer@freebsdjournal.com
- Art Director** • Dianne M. Kischitz
dianne@freebsdjournal.com
- Office Administrator** • Michael Davis
davism@freebsdjournal.com
- Advertising Sales** • Walter Andrzejewski
walter@freebsdjournal.com
Call 888/290-9469

FreeBSD Journal (ISBN: 978-0-615-88479-0) is published 6 times a year (January/February, March/April, May/June, July/August, September/October, November/December).

Published by the FreeBSD Foundation,
PO Box 20247, Boulder, CO 80308
ph: 720/207-5142 • fax: 720/222-2350
email: info@freebsdjournal.org
Copyright © 2015 by FreeBSD Foundation.
All rights reserved.

This magazine may not be reproduced in whole or in part without written permission from the publisher.

LETTER from the Board

FreeBSD is heavily used in the Enterprise, which is the theme of two feature articles in this issue. Sean Chittenden of Groupon and Benno Rice of Isilon detail their companies' unique experiences with FreeBSD. As Isilon and Groupon have built quite different systems with FreeBSD—the former, a clustered storage device, and the latter, a web service—you'll find a wealth of information in these two articles.

While it is important to work on a rock-solid operating system, that is still only a small part of the software required to have an interesting and usable system. The amount of code maintained in software packages far outweighs that within the operating system, and it is the packages that most users interact with on a daily basis. From web browsers to web servers, development tools to graphical editors, the package system is how the FreeBSD community gets access to tens of thousands of pieces of open-source software. In the second issue of this *Journal*, Bryan Drewery wrote about the updated package system when it debuted. Now he's back to tell us about lessons learned and how these lessons have helped shape pkg as it is today.

Some newcomers to the FreeBSD Project only know about us through our code base, but supporting that code base requires not just engineering but also a lot of other things. The FreeBSD Foundation supports the Project by sponsoring conferences, providing legal support and trademark protection, and generally handling all the details that many of us, as software developers, simply take for granted. In our continuing section on "Interacting with the FreeBSD Project," Deb Goodkin, the Foundation's Executive Director, takes us inside the Foundation and shows us what the Foundation is doing today and describes its plans for tomorrow.

2015 has been a banner year for FreeBSD and BSD-related conferences; in fact, there are so many that I can't make it to all of them. From AsiaBSDCon in Tokyo, to BSDCan in Ottawa, VBSDCon near Washington, DC, EuroBSD in Stockholm, and BSDCon Brazil in Fortaleza, FreeBSD truly spans the globe. Whenever the FreeBSD Foundation funds someone's attendance at one of these conferences, they ask the attendee to send in a report of their experience, and this month we are publishing Shonali Balakrishna's report from BSDCan 2015.

Rounding out the issue, we have the *Journal's* regular slate of columns: Frederic Culot with Ports Report, Greg Lehey with a book review of *The Practice of System and Network Administration*, Glen Barber with svn Update, and Dru Lavigne with "This Month in FreeBSD" and the Events Calendar.

Enjoy!

Sincerely, George Neville-Neil

For the **FreeBSD Journal Editorial Board**

Groupon[®]'s Deal on FreeBSD

Familiar look and feel,
slightly different trim,
now with enhanced results.

By Sean Chittenden

It's 7:27a.m., I'm queued up at a metering light on my way to work, and my phone lights up with a message from one of database team's senior engineers:

"Do you have some time for debugging session? We most likely have a BSD issue."

I grimace and start to think about what he could be stuck on. I'm not exactly sure where their team is in the execution of their current project either, so I can't easily speculate. The thing is, I don't normally hear from this guy unless he's really stuck on something. So hearing "a BSD issue" when I'm not even halfway through my first cup of coffee isn't giving me a warm and fuzzy feeling, and it is definitely not how I want to start my day.


My calendar is jammed full of meetings including a presentation and two deliverables, not to mention the odd escalation that shows up and requires immediate attention. But a "BSD issue" sounds like a rabbit hole of an issue and oddly specific.

I fire back, "I'm about to get on the freeway, can you call me?" He responds, "Can you get back to me when you are in front of `tmux(1)`?"

"Uh oh," I think, "this is probably something legit."

It's been a few months since we've run into any issues, so maybe we're due for an OS bug. Operations is in the process of finalizing its latest iteration of a database containerization platform, so hearing we've stumbled across something doesn't shock me since we're still working





through some details of the current iteration. After all, we were playing with **VIMAGE** at one point so it's not out of the question. Further, with over five petabytes of storage on the most recent deployment, it wouldn't surprise me to find out we've also just run across something run-of-the-mill this morning that comes with having a large deployment.

As I'm rounding the last corner on my way to the office, I get a text message from Chris Schneider, manager of the global database team, "Hey, Bob needs help and thinks he has a BSD issue because he's seeing a `listen(2)` queue overflow kernel message when debugging the new service checker daemon."

I think, "Whew! New code and it's in the application's network code, this should be easy."

And that's how it goes. Every few months as the footprint of FreeBSD grows in the organization, we collectively stumble across "new and exotic error messages" that we add to our repertoire and around which we build processes and understanding.

Background

A little over a year ago Groupon began developing a next-generation database platform aimed at increasing reliability, reducing the cost of operations, and adding resiliency to some of the anticipated changes in the storage industry, notably the proliferation and widespread adoption of solid-state drives (SSDs). When the database project was initially started, my boss directed me to use SSDs because he wanted to get away from some of the more expensive flash memory technologies we already had in house. While the legacy flash hardware had been adequate for years, its integrated OS drivers and hardware have been operationally problematic and the cost made it prohibitive to reasonably deploy throughout the entire database tier. So off to work we went.

When building out large footprints, the law of large numbers is not just a convenient theorem to cite in a discussion; it needs to be a fundamental design consideration. The words "shouldn't happen" are taboo and a clear sign more research and analysis are required. Like any other part of business, identifying both strategic and operational planning assumptions and managing the risk/benefit is critical. If, however, businesses do manage to incorporate the entirety of potentially known issues or "risks" into their planning assumptions, they're skilled, a bit lucky, and a little clairvoyant. If the planning assumptions don't identify *and* consider all the known risks or create the corresponding contingency solutions to make the risks acceptable, there had better be good justification. In this case, the level of planned consolidation (>15:1 reduction in hardware) and operating efficiency gains that would come from moving off of spinning media and over to SSDs was clearly a benefit; however, the quantity of flash memory cells and the fidelity of the data were identified as risks stemming from the change in storage media. We knew this up front so we had to do something from the onset to manage the identified risks.

In Operations some of our primary planning assumptions include: a percentage of our servers die, hard drives fail, RAM develops single-bit parity errors, and datacenters go dark. We understand and plan for all these situations—spare servers, use of hot-swap hard

GROUPON's Deal on FreeBSD

drive bays, procure ECC RAM, and geographic redundancy initiatives—and much, much more. The introduction of SSDs (versus spinning rust) introduced a substantive change to the operating model because the flash cell-based SSDs have a dramatically elevated bit error rate (BER). We knowingly understood this change in storage media would result in a material increase in the rate at which data would corrupt—rot—at rest. In the case of bit rot, the law of large numbers dictates it's not a matter of "if," but "when," and "how often." This created the need to update our team's planning assumptions and risk mitigation strategies.

The performance gains of moving from spinning rust to SSDs are significant. Seek times on 10K RPM disks are on average measured between 2 and 5 milliseconds, but the 99th percentile latencies can be measured in the tens or hundreds of milliseconds. Moving from milliseconds ("ms," one thousandth of a second) to microseconds ("us," one millionth of a second) represents a three order-of-magnitude improvement and this performance gain is now able to be rolled out to all teams and applications, not just the Tier-1 databases. And sure, the performance gains are nice for benchmarks, but for

engineering teams, performance gain translates into a real-world efficiency gain which shrinks development schedules. Inefficient queries that were frequently cache-miss and would take 50–100ms now take 60–200us. CPU usage rises to desirable levels and engineering teams don't have to worry as much about performance efforts. In effect, use of SSDs allows us to trade OpEx for CapEx by reducing the time required to ship many products, but what's the trade-off and what does an organization do about it?

A year before Carnegie Mellon University and Facebook pointed out in their fantastic study published at the ACM SIGMETRICS '15 conference in June of 2015, "A Large-Scale Study of Flash Memory Failures in the Field," we were toiling away with designs for the database platform that would compensate for our observed bit rot on SSDs. At the time, we were seeing errors, but we didn't know how widespread the problem would be, just that we expect it to happen and therefore we had had to compensate for this inevitability. When the CMU/FB flash memory paper came out, their findings were very much in line with what we had observed; however, their testing methodology was much more extensive and the results poignant. To sum-

ISILON The industry leader in Scale-Out Network Attached Storage (NAS)

Isilon is deeply invested in advancing FreeBSD performance and scalability. We are looking to hire and develop FreeBSD committers for kernel product development and to improve the Open Source Community.



We're Hiring!

With offices around the world, we likely have a job for you! Please visit our website at <http://www.emc.com/careers> or send direct inquiries to karl.augustine@isilon.com.



EMC²

ISILON

marize a few highlights regarding their findings:

- six different major SSD vendors
- uncorrectable bit error rate (UBER) is reasonably common
- the most reliable storage platform saw at least one UBER on 4.2% of its installation over a 12-month period of time
- the least reliable platform saw at least one UBER on 34.1% of its installation over the same duration and workload

Clearly it's still early days for SSDs and things will improve over time, but the variance in UBER across vendors is worth noting. For one workload, we selected a vendor's platform and experienced a >15% UBER in under a four-month period of time and expect that to settle out around 30–40% over a similar 12-month duration. One nonscientific observation we have made is that drives which throw UBERs will throw lots of errors and that the distribution of UBERs is not even across the fleet (i.e. some drives throw lots of errors, others throw very few, or no errors – we have not looked into why as of yet).

Maybe you are thinking to yourself, “So what if a bit flips? It's one bit. A gigabyte is 8-billion bits and one bit "" shouldn't actually affect anything.” Out of 8-trillion bits, one bit is vanishingly insignificant, yet if a flipped bit happens to be in `/usr/local/bin/vim` and it refuses to start, is it still insignificant? What if the bit flipped in the middle of a database table, how would you know? The `vim(1)` example is not hyperbole, as it was the first program to experience a detected UBER. The error occurred three months after one of our test machines was provisioned, it's `mtime` was three months in the past, and one day `*BAM*`. That's basically how bit rot happens: it silently, quietly, and probabilistically kills a bit somewhere in the middle of the night, and does so without any fanfare or helpful error messages. Try playing around with striped disks instead of mirrors in a test environment for a few months. On a disk of full of 1's and 0's, suddenly one of them flips. It's possible someone else on this planet is more unlucky than you and you won't have any errors, but that level of denial won't change reality. Sweet dreams, everyone.

Assuming you didn't pick the platform which had just over a third of its drives throw UBERs, what is more concerning was the incidence of UBERs increased in proportion to the utilization of the SSD. At the time of this publication and according to the CMU/FB flash memory paper, the probability of an SSD from one of the reliable vendors experiencing at least one UBER over the course of a 12-month duty cycle will stabilize

between one in 10 or one in 25. Depending on the drive density in a server, that could be >1 UBER per server per year if you load up 24 SSDs per server, or just over one in 10 servers if you only put 2x SSDs per server. Are we feeling squeamish about using SSDs in production yet? If not, I encourage you to find the CMU/FB flash memory paper and ruminate accordingly because those error rates are too high to simply ignore.

Let's put it another way: if bit rot represents a

One of the other interesting findings from the CMU/FB flash memory paper was contained in section 4, notably the period of early failures on a 720GB SSD starts to wane around 3TB of written data. Taking RAID into consideration and only 8x SSDs per host that would mean the current generation of SSDs should have a burn-in procedure write out just over 24TB worth of data before putting the host into production. As the technology becomes more robust, I'm not sure this advice will remain accurate.

needle in a haystack, is one needle in every other bale of hay acceptable? How about one needle in every two hay bales? What's the probability of a flipped bit causing a problem? How about a dozen new needles every quarter added to every hay bale? Would anyone notice or care? Does your organization's appetite for risk change if we change the haystack metaphor from a sewing needle to a used hypodermic needle from an adjacent clinic? Thanks, but no thanks. Ignorance is not bliss and we want certainty at the storage layer for our databases, not chance and luck.

To put some context on the operating circumstances, Groupon's core business is to provide a commerce experience which delights our customers. The business is such that we require operational efficiency, reliability, and correctness. In the previous 12 months ending in Q1 2015, Groupon had \$6.3B in gross billings. In Q1 of 2015 alone we sold 54M units (units reflect vouchers and products sold before cancellations and refunds), and have cumulatively sold over 800M units as of Q1 2015. At the end of Q1, we averaged over 160M monthly unique visitors, and better than 80% of our customers return to the site to make future purchases (ForeSee Groupon Customer Satisfaction Study, March 2015—commissioned by Groupon).

A significant portion of that business was going to land on this new platform so correctness wasn't something we could be uncertain about. We needed to plan for the future and accept that when handling large amounts of data, whatever could theoretically happen, will happen, and will happen at a predictable rate. To guarantee the fidelity of information on

Groupon's Deal on FreeBSD

petabytes of data across arbitrary applications that may or may not have the ability to checksum their own data, we needed some way of mitigating the identified risks from SSDs.

The solution and outcome should be obvious at this point, but the unexpected benefits along the way—both technical and organizational—have been numerous and remarkable.

FreeBSD to the Rescue

There are many technical problems in the world, but many of the actual hard problems aren't technical so much as they are organizational. The primary problem we needed to overcome was the ability to run arbitrary applications that had no capabilities to checksum their own data at rest, and this technical problem required a technical solution. In 2014 Groupon was a Linux-dominant shop. After much cantankerous wrangling, discussion, review, and testing of alternatives, we settled on FreeBSD with ZFS as the technical solution to the problem.

In the world of mature operating systems with a great track record in production, it's hard to beat FreeBSD. Yet after we settled on and committed to FreeBSD as the solution to our imminent and burgeoning bit error rate ("bit rot") problem, the next question was, "But will it blend?" We had secured a means of solving our bit rot problem, but we weren't sure how the rest of the organization would adapt to an OS that wasn't Linux.

Technically what does it mean to change or grow support for an additional operating system? It turns out, not that much if you're set up to handle multiple distributions of Linux. "So this project is going to support current growth, integrate previous acquisitions, plan for future application and product launches, AND change operating systems for the database tier?" "Yup, because the change isn't that significant." Sure, it requires some investment up front, but the analogy is closer to owning a Mercedes and buying a BMW and learning to drive the new car effectively. The shifter is in a different place, the radio is different, but both are fantastic pieces of engineering, require gas, and get you to your destination comfortably and in style.

No `fsck`'s Given

DTrace support, along with the ability to easily roll a custom kernel and use of `poudriere(1)` were some of the initial big ticket items that showed up in the pro column when making the decision. Without much fanfare, however, the

lack of file system checks `fsck(1)` showed up as a strength, too, and now it's one of the bigger items that no one misses or even talks about anymore. Imagine running a `fsck(1)` on a near line backup server with 288TB of storage. A traditional block-oriented file system would have taken days or weeks to complete. With ZFS? Zippo. Literally, no time. Such activities and concerns are now anachronistic in our modern environments and no one has waxed poetic about their absence. Though at the time ZFS was introduced, it was a question that frequently came up because file system checks are one of the tools administrators can use if they get into trouble. Taking away a utility as ingrained as `fsck(1)` highlighted one of the challenges of organizational change: personal anxiety from either learning something new, or trusting something new in production needs to be considered.

During the lead up to committing to this direction, it was clear there was some apprehension with some of the database administration team. FreeBSD isn't Linux, therefore it's different, but how different? Supporting a new OS isn't necessarily about making sure all programs compile and run. Is FreeBSD different in a material way? Is the transition going to be an overwhelming experience? What is the level of effort required for each admin to become familiar with the new OS? When will the uncertainty or doubts subside?

The answer in our case was to conduct a series of 30-minute, online tutorials and open mic question-and-answer sessions. In order to meet the scheduling needs of a worldwide team, we scheduled video conference calls every few days at 6:30 a.m. PDT to meet and go over some aspect of FreeBSD. To keep the sessions open for candid conversation and focused on the learning needs of each person, we broke out and grouped attendance based on skill level and identified personality traits that kept the open mic Q&A productive and bidirectional for everyone.

The first session was as simple as logging in via `ssh(1)`, logging in to MySQL and PostgreSQL, running a few basic observability commands such as `top(8)`, `iostat(8)`, `vmstat(8)`, and a few new commands such as `sysstat(8)` and `gstat(8)`. The overwhelming response was, "This isn't different from what we're used to." To finish, we revisited `top -m io` as a "one-more-thing" moment, and that yielded a number of, "OH! You mean I don't have to XYZ anymore?" The unknown had become exciting and now helpful to everyone's day-to-day activities.`

Sessions after that moved quickly and covered things like the base system versus `ports(7)`, using `pkg(1)`, compiling a port by hand, and the difference between `/etc` and `/usr/local/etc`. Shortly after, we went in to some of the more murky waters like `pf(4)` and `ipfw(8)`, and the history for why there is more than one firewall in FreeBSD. During a few sessions a little DTrace-based observability trickled out here and there in order to show some latency quantiles, counts, stack traces, etc. The intention wasn't to teach DTrace now so much as give perspective and instill the understanding that "if all else fails, there's always DTrace." And if it does come to using DTrace in production to identify a problem, chances are you're not having the least stressful day of your life and other people will be around helping out.

It wasn't until around the fourth session that we introduced ZFS. In that seminar we covered snapshots, cloning, `zpool scrub`, and rolling back. To punctuate one of the more profound uses of ZFS, we had also replicated a small 1TB database in advance. During the Q&A, we took a ZFS snapshot without shutting the process down, trashed 600GB worth of data with a mix of faux-inept `UPDATES`, `DELETES`, and `DROP TABLES`. We then shutdown the database and rolled back to a previous snapshot in about 30 seconds. When we brought the process back up, it had about 5 minutes worth of replication lag to catch up on. Compared to other snapshot technologies which incur a performance hit when taking them, ZFS was like magic.

TIP: Use DTrace during a 'zfs snapshot' to measure VFS write latencies. Wow people with write latencies measured in single-digit microseconds (not milliseconds) ([see Box 1](#)).

Not bad for spinning rust. Did someone forget to mention ZFS can also be really, really fast?

Everyone's personal rate of adoption and rate at

which they address the anxieties from using something new is different, but there's clearly a trajectory for anxiety where it moves from the dread or excitement of learning something new (frequently both), to some feeling of unease due to a lack of knowledge or experience, acceptance and embracing the new, and eventually ends when the day-to-day activities that were semi-unsettling become natural and comfortable. Changing technology is not entirely about the technology, it's about the process of working with people and building the necessary familiarity required to scale a technology in an organization.

TIP: Do benchmarking and other high-glitz testing to prove systems are not fragile while working to buttress confidence and combat anxieties. For instance, take a random write workload from a Linux host with a Fusion IO card and move it to FreeBSD, ZFS, and give the zpool only 16x spinning disks. Suggestion: repoint a stream of >100K ganglia metrics being written to RRD files ([see Box 2 next page](#)).

Within a week most everyone's personal Rosetta Stone for translating between FreeBSD and Linux was 50% complete, and in another two weeks it was closer to 80%. Everyone became pretty confident their muscle memory would work equally well under pressure on FreeBSD as well as on Linux. As it turned out, FreeBSD is a fantastic crossover platform for picking up ZFS and DTrace without having to mentally remap basic commands like `top(1)` to `prstat(1)`. The lack of friction for the basics is one of the things that made this transition possible. The oddball question about file name limits for includes in `pf.conf(5)`, or other elements of the stack get escalated up, but they're usually a by-product of making aggressive use of the various functionality in FreeBSD.

Coincidentally the start of this project coincided with a rash of OpenSSL vulnerabilities, so the

```
# dtrace -s vfs-io-postgres.d
Latencies (ns)
postgres Write
```

value	----- Distribution -----	count
1024		0
2048	@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@	1325
4096	@@@@@@	267
8192	@@	72
16384		0
32768		0
65536		19
131072		2
262144		0

1


```
# zpool iostat tank 1
```

pool	alloc	capacity		operations		bandwidth
		free	read	write	read	write
----	----	----	----	----	----	----
tank	958G	9.94T	0	210K	1022	330M
tank	958G	9.94T	1	207K	4.99K	326M
tank	958G	9.94T	32	30.5K	79.9K	46.9M
tank	958G	9.94T	22	9.62K	202K	15.9M
tank	958G	9.94T	15	10.2K	169K	16.5M
tank	958G	9.94T	36	10.5K	198K	14.9M
tank	958G	9.94T	6	10.8K	39.4K	17.4M
tank	958G	9.94T	12	189K	209K	298M
tank	958G	9.94T	1	210K	7.96K	340M
tank	958G	9.94T	10	218K	23.0K	355M
tank	958G	9.94T	2	224K	4.49K	359M
tank	958G	9.94T	6	228K	12.5K	367M
tank	958G	9.94T	7	140K	53.4K	225M
tank	958G	9.94T	9	26.9K	40.9K	44.0M
tank	958G	9.94T	0	9.43K	0	13.9M
tank	958G	9.94T	0	9.69K	0	16.3M
tank	958G	9.94T	1	74.0K	3.49K	120M
tank	958G	9.94T	6	226K	17.0K	366M
tank	958G	9.94T	0	225K	0	385M
tank	958G	9.94T	0	176K	0	515M
tank	958G	9.94T	0	84.7K	0	382M
tank	958G	9.94T	0	39.6K	0	163M

top “just in case.” Statements like “Whoa, we couldn’t do that before” were routine for a period of time; then everything settled down to a steady state of productivity and enhanced use of the platform. We actually found a good, sanctioned use of ZFS dedupe (versioning historical packages), using `carp(4)` on top of `vlan(4)` tagged LACP `lagg(4)` interfaces became “the way,” etc. With `lz4` compression we had more than doubled our effective storage capacity per server (an important thing when putting large numbers of SSDs into production), yet months after the initial testing, deployment, and adoption, we still hadn’t seen the boogie man of bit rot.

2

ability to build and deploy updated packages quickly became one of the more welcoming features. In a little over 40 minutes a full rebuild of every package in use, with our own custom patches, was made available for installation worldwide. Twiddle a few knobs in a half-dozen files, and you could integrate custom in-house software, each package built in an isolated clean room `jail(8)`, complete with a good chain of trust all the way back to the original package

3

```
$ zpool status tank | head -n 3
pool: tank
state: ONLINE
scan: scrub repaired 4.50K in 53h44m with 0 errors on Tue May 26 21:36:26 2015
```

Waiting in the Dark

The vision of the database project was this: build a database platform robust to bit rot that the database team could administrate. The second half of that objective had been achieved, but it wasn’t until a few months after the first set of systems were provisioned that we had our first catch. Ever since, every few weeks we see ZFS

author. This was an off-the-shelf win that came by simply deploying `poudriere(1)`, something we completed in a matter of days. Ever since, we only spend a few hours every month maintaining a full release and this agility has been noticeable. As one engineer put it, “`poudriere(1)` is a thing of beauty, something to be admired.” The package building infrastructure sets FreeBSD apart compared to other maintained platforms.

Culturally we’d rounded the corner with the flow and ebb of anxiety, people had adjusted and were now flourishing. General concern for FreeBSD in production had waned and we no longer felt compelled to travel places with a lap-

detecting and repairing errors (see [Box 3](#)). Our first “catch” happened months and months earlier and every time someone notices, we get to say, “So that just happened,” and move along with life without caring. 4.5K repaired? Probably a single bit flipped. 45M repaired? A vendor’s SSD controller probably repeatedly crashed between scrubs and wrote out a reasonable amount of bogus data. We can do all of this because ZFS can detect corrupted bits and self-heal for well-constructed zpools where enough redundancy exists. Automatically. At runtime. Transparent to the application.

We do scrub zpools, and some take ages to

```
pool: tank
state: ONLINE
scan: scrub in progress since Mon Jun 14 04:30:54 2015
      6.14T scanned out of 108T at 16.8M/s, (scan is slow, no estimated time)
      0 repaired, 5.71% done
```

4

complete. Big bricks take weeks to complete (see Box 4). And others with SSDs still take a moderate amount of time to run (see Box 5).

There is no mission-accomplished moment for this effort. It's just a vigilant, constant process of iteration, reaction, optimizing for the future, and increasing the rate at which we can iterate and improve the current situation. But does it mean we've been successful in mitigating the technical risks? Yes. A resounding yes. The rate of iteration started slowly as all of the tooling took hold, but now that we're on to our third, arguably fourth iteration, the duration between iterations is getting smaller and the improvements per iteration are increasing. On the technical side we're using `poudriere(1)` to great effect, deploying databases using `iocage(8)`, moving to DNS-based failover mechanisms which will scale bet-

ter than prior layer 2 efforts using `carp(4)`.

And that "BSD issue" from earlier? It turned out to have been an issue with the service check script, kinda. FreeBSD doesn't require you to troll through ``netstat -s`` to find the overflow counters; instead, the kernel emits a helpful error message that informs administrators the `listen(2)` queue of `un-accept(2)`'ed filedescriptors had overflowed. During development Bob had stuck a 60 second `sleep(3)` call in the request handler to test something out, which is fine, except the load balancer was hammering away on the service checker daemon every second, and with a `listen(2)` backlog of eight, it

5

```
% zpool status tank
pool: tank
state: ONLINE
scan: scrub in progress since Tue Jun 16 15:52:26 2015
      1.58G scanned out of 1.17T at 5.56M/s, 61h9m to go
      0 repaired, 0.13% done
```

RootBSD

Premier VPS Hosting

RootBSD has multiple datacenter locations,
and offers friendly, knowledgeable support staff.

Starting at just \$20/mo you are granted access to the latest
FreeBSD, full Root Access, and Private Cloud options.



www.rootbsd.net

GROUPON's Deal on FreeBSD

took exactly eight seconds before the kernel started blasting out informational messages. The fix was easy and took 20 minutes to add: mixin a multithreaded socket handler and shuffle around a few method calls to allow multiple threads to participate in `accept(2)`'ing new connections off of the `socket(2)`. Sixty minutes from open to resolved. An easy problem that didn't actually require escalation, but a subtle difference and a reminder of the consequence of a dominant monoculture mindset.

Introducing technical changes to an organization requires successfully navigating the necessary personnel change elements. Taking into consideration the needs of individuals who will be long term participants or owners of the resulting technical change should be a primary objective of the entire change process. People going through this change process will transition through a series of phases before becoming fully aligned with the change. This is especially true when not all beneficiaries were participants in

the research or decision making process (this frequently happens in large organizations). The model we worked from was centered around addressing the announcement and introduction, personal anxieties, building technical and organizational acceptance, followed by achieving steady productivity and operational gains. Consider bringing in external training to fill gaps in understanding (many of which may transcend the particulars of a specific OS, we did that for a broader audience and it worked out well - thank you Rich and Dru), set up small-group seminars, and even share "war stories." Once the organizational changes were considered and taken into account, the fruits of the technical change were compelling for us and the extra tool in the toolbox has proven to be valuable and has mitigated the risks identified in our planning assumptions (see Box 6). Being able to do performance monitoring for IO in the microsecond level and having automatic self-healing from bit rot? Yes, please. Oh, and sleep well. We do. ●

FreeBSDTM JOURNAL NOW AVAILABLE AS A DYNAMIC EDITION!

The Dynamic Edition format offers subscribers the same features as the App, but permits viewing the Journal through your favorite browser.

Read It  Today!



The DE, like the App, is an individual product.
You will get an email notification each time an issue is released.
A one-year subscription is \$19.99, and a single copy is \$6.99—the same pricing as the App version.

www.freebsdfoundation.org



```
#!/usr/sbin/dtrace -s

#pragma D option quiet
#pragma D option bufsize=8m
#pragma D option switchrate=10hz
#pragma D option dynvarsize=16m

/* See /usr/src/sys/kern/uipc_mqueue.c for vop_read_args.
 * Also see sys/uio.h.
 */

dtrace:::BEGIN
{
    i = 60;
}

profile:::tick-1sec
/i > 0/
{
    i--;
}

profile:::tick-1sec
/i == 0/
{
    exit(0);
}

vfs::vop_read:entry, vfs::vop_write:entry
{
    self->ts[stackdepth] = timestamp;
    this->size = args[1]->a_uio->uio_resid;
    this->name = probefunc == "vop_read" ? "read" : "write";
    @iosize1[execname, this->name] = quantize(this->size);
}

vfs::vop_read:return, vfs::vop_write:return
/this->ts = self->ts[stackdepth]/
{
    this->name = probefunc == "vop_read" ? "read" : "write";
    @lat1[execname, this->name] = quantize(timestamp - this->ts);
    self->ts[stackdepth] = 0;
}

profile:::tick-15sec
{
    printf("--- Tick 15 -----\n\n");
    printf("Latencies (ns)\n\n");
    printa("%s %s latency (ns)\n%d\n", @lat1);
    printf("IO sizes (bytes)\n\n");
    printa("%s %s bytes\n%d\n", @iosize1);
    printf("-----\n\n");
    trunc(@lat1);
    trunc(@iosize1);
}
```

SEAN CHITTENDEN is an Architect for Groupon Production Operations (seanc@groupon.com). He is a long-time participant of the FreeBSD (seanc@FreeBSD.org) and PostgreSQL communities, a 15+ year veteran of large scale web infrastructure including databases, networking, and storage. In a prior life he owned and ran a datacenter and technology reseller whose clients included Facebook and Yammer before becoming a ghost in the machine among Silicon Valley startups.



FreeBSD: THE ISILON

Once upon
wanted a
gigaby
Tr
o

EXPERIENCE

a time a company called Isilon was founded by some people who
a way to build a storage array that was not only easy to expand in
ytes but also easy to expand in IOPS.

Additional storage arrays tended to be built from some kind of head unit,
optionally with a hot-, cold-, or warm-spare, and a bunch of shelves full
of drives. This was the traditional block storage model. When newer
file-oriented products like NetApp arrived, they copied this model.

The problem was working out how to scale it over time.

Scaling storage capacity (gigabytes) was easy, up to a point. You added more drives to your shelves, and, as they filled up, you added more shelves until you ran out of controller ports on your head unit. At that point you had to upgrade your head unit and suffer downtime. The other scaling axis was your ability to get data off the drives and out to clients (IOPS). Again, the head unit model tended to hit a wall and require a downtime to replace the head unit with a more powerful one.

The key distinction of the Isilon model was to shift from a head-unit-plus-shelves model to a cluster of mostly redundant nodes model. Each node would be a fully capable, mostly stock-standard, server-grade system containing CPU, RAM, NICs, and drives. Each node would cooperate with the other nodes in a cluster sharing a single file system, and, as each node was added, not only the storage capacity but the serving capacity increased. The other advantage this model gave was resilience. The cluster could place files in such a way as to protect against drive or even node failures. This gave better utilization of resources as in the head-unit-plus-shelves model, you either had to

have redundant capacity sitting idle or lose capacity when one head died. The product was sufficiently compelling that EMC acquired Isilon in 2010 and Isilon has gone on to provide excellent returns on EMC's investment.

What's this got to do with FreeBSD?

Well, each of those nodes that Isilon sells, and they've been selling for 10 years or so now, runs FreeBSD. It's not stock FreeBSD by a long way, but the OneFS operating system, as it's known, is built on FreeBSD and provides us with a solid foundation and scaffolding on which to build our product. Obviously we're not the only ones to take advantage of this, as it even seems like a large proportion of the storage industry uses FreeBSD, at least the file-oriented parts. What we'd like to show in this article is how we've used FreeBSD, where we've hit problems, what we're now doing better, and why we think we've hit on a good model for other companies building Enterprise-grade products on top of FreeBSD.

THE PAST

The first version of OneFS was built on top of FreeBSD 5, tracking what is now the stable/5 branch in Subversion. Of course it was all CVS

FreeBSD: THE ISILON EXPERIENCE

back then. There were projects that upgraded our underlying FreeBSD code to stable/6 and then stable/7 and the version control system in both cases switched to Subversion, but during the merge to stable/7 things didn't go exactly according to plan. There were a bunch of issues that occurred during the project to update the version of FreeBSD underlying OneFS to the stable/7 branch and the end result was that we got gun shy. The FreeBSD upgrade projects started to look risky. The end result was that we stayed on stable/7 far, far longer than we should have.

The primary reason we ran into problems during the stable/7 merge was that our modifications to FreeBSD were extensive, and, unfortunately in some cases, not well marked. This led to large swathes of change we had to merge every time we switched to a new stable branch. Each one of these changes brought with it the potential to be lost, to be mis-merged, or to break something in some other way. One of the big scares that happened during the stable/7 merge was an important internal change that got missed and was only found at the last minute. Things like this, along with the general arduousness of the merge projects themselves, led to a level of complacency.

It can look tempting to not upgrade: you don't have to worry about lost changes, you don't have the risk, and you can spend your valuable effort-hours on other projects. The problem is that as time goes on, you end up lacking the advances that the upstream code-base is making. Worse than that, you lack the fixes that are coming in. As stable branches roll out of active maintenance you can end up in the position where you essentially are the sole "owner" of code you could be sharing with others. To me, this is an insidious side of FreeBSD's release model. Eventually we realized we couldn't stay on stable/7 forever.

Organizations that want a "safe" FreeBSD are encouraged by the naming (it says stable right there in the branch name!) to use the stable branches. The problem is that once there, you face a merge/upgrade project to get to the next one. And the next one. And the next one. This puts a brake on any desire you may have to do the upgrade and tempts you into staying on that branch for just a bit longer. And then the stable branch you're on isn't actively maintained anymore. And suddenly you have to spend those effort-hours you saved on diagnosing and fixing bugs in code that nobody upstream is interested in anymore. It's a classic technical debt situation.

Fortunately, staying on the stable branches isn't the only option.

When I joined the Isilon division in 2013, we'd decided that if we wanted to get any more performance out of our nodes, we needed to revamp a bunch of things in the FreeBSD kernel, primarily the VM and in various aspects of how data travelled to and from drives. Some of this had already been done in upstream FreeBSD and this led us to the obvious conclusion that it was finally time to get off stable/7. The obvious path was to move to stable/9, which was the top stable branch at the time. However, we thought back to the previous issues with this model and came up with a new idea.

We were going to move to head.

This felt, initially, really scary. Head is supposed to be the uncharted country full of bugs that will jump out and ruin your life. The reality though is that FreeBSD's head has been relatively stable, modulo some wobbles, for a very long time. It certainly churns faster than the stable branches, but the days when you had to pick which revision to install lest you end up with an unbootable system have, over time, been replaced with the situation that things are generally safe provided you avoid some known bad revisions.

The reality is that FreeBSD's stable branches are an artifact of its release model, and FreeBSD's release model is largely irrelevant to us. FreeBSD, to Isilon, is a set of code that we consume to form the base of the software we run on our product. We don't need it to be at a particular point in its release schedule as long as the code we get is stable. The key realization was that what we wanted out of a stable branch was stability. We wanted an absence of bugs. The problem was that every stable branch merge project caused bugs. So were we gaining anything? In the end, we decided that the correct path to comfort wasn't to avoid bugs by sticking to stable branches; it was to concentrate on QA and testing to make sure we felt comfortable that we would find the bugs and fix them. This would benefit us, in that we could do more work directly with FreeBSD and have it show up in our releases in a reasonable time frame, and benefit FreeBSD, in that we'd be finding and fixing bugs and developing new stuff. All of this sounds awesome; the main problem was how to get there.

THE PRESENT

So now we were in a position where we knew where we wanted to be; we just needed to

work out how to get there. In theory this was the same kind of project we'd done before, but instead of crossing from one stable branch to the next we were moving from stable/7 to head, skipping over stable/8, stable/9, and, eventually, stable/10 in the process. This was not a small project.

The original lead of the project took a rather ambitious approach. Their plan was to take a clean checkout of FreeBSD head and effectively rebuild our entire set of modifications onto it as needed. This had the advantage that the end result would remove any modifications we turned out to no longer need. It had the disadvantage that it was a lengthy process and required a lot of upfront guesswork to know what was actually needed and what wasn't at which point. This was the process in effect when I joined. Eventually another engineer who'd run a previous stable merge project joined and proposed a new method.

This method was almost the inverse of the previous one. The new plan was to work out which FreeBSD head branch Subversion revision we could reasonably be sure of having merged up to. We then started applying FreeBSD Subversion revisions to head sequentially, sorting out conflicts as we went. The first issue we had to overcome was that Subversion won't track merge metadata between repositories. We overcame this with a Python script that could manage a Subversion property containing our own merge metadata. The other issues were bigger.

As noted, we were on stable/7. We were applying revisions from head. These didn't always line up perfectly and so the first weeks of this project were spent ironing out a large number of conflicts. Additionally this process was very single-threaded. Only one person could be merging at a time. We got around this by exploiting the fact that I was based in Melbourne, Australia, at the time and the other engineer doing this work was in Santa Clara, California. This meant we only had an hour or two of overlap in our workdays which we could use to sync up and hand over.

One interesting thing about this approach was that instead of effectively starting over with a clean FreeBSD and building a OneFS on top of it, this started with OneFS and tried to incorporate the changes in FreeBSD back onto it. This meant that our testing infrastructure could continue to work. Or it did until some changes to the network stack in FreeBSD 8 came in and blew apart our Infiniband stack. This took months to resolve, due to a lack of person-hours rather than anything else, and slowed us down tremendously.

In the end this approach proved to also be too slow. At this point we'd missed being included in the release we'd been aiming for. We went back to the drawing board again.

One of the problems we'd been facing in the first approach was the sheer amount of churn we had to look through between our stable/7-based code and FreeBSD head. It was very hard to look at a given diff and work out whether it was our change or a FreeBSD internal change. Some of our changes were well marked but a lot weren't. However, the work we'd done with merging FreeBSD revisions in the second approach turned out not to have been wasted. It had gotten us up to, approximately, FreeBSD 8 and in the process had also removed a lot of our variance to the head line of changes vs the stable/7 line of changes. This meant that we could run a diff from the tree we'd used for our second attempt against FreeBSD head at about the same revision and the resulting diff was mostly our changes. Thus was born the final approach.

The approach that succeeded involved taking the diff we'd generated and spending a few weeks working out the logical components inherent in it. Most of our changes tended to be for specific purposes, whether for adding stuff to the buf handling, the filesystem I/O path, or tuning certain parameters, or adding extra debugging support. Once we'd worked out roughly which bits of the diff corresponded to which logical groups of changes, we could assign each logical change group to someone, and thus many people could be working in parallel.

We started with a clean checkout of FreeBSD head, similar to the original approach, but then each logical change group was merged in and the resulting code tested using a mixture of the Kyua/ATF tests that are present in FreeBSD along with some extras we developed internally plus a set of other basic tests we could run on something that was, at that point, a heavily modified FreeBSD. Once enough parts were connected, we started bringing our OneFS files system code online and then the supporting userland pieces. Once we got everything in place so that we could run our internal automated test lists, we then set about fixing all the bugs.

The above makes it sound simple, but it wasn't. The project started early in 2014. We weren't able to mount the OneFS file system until late August 2014. We had our first automated test run in mid-September. We merged to the OneFS head branch on April 1, 2015. It was the culmination of a ton of work from a very dedicated group of people and I'm really proud of what we achieved.

So what next?

THE FUTURE

Well, obviously we want to make sure we never have to do a project of this magnitude again. We've already got a process sketched out such that all the changes from where we stopped merging to an appropriate point in head will end up merging into each release as it comes through. We put in the effort to get this far, and we don't intend to get behind again. We're also looking at more ways to contribute to FreeBSD.

...the culmination of a ton
of work from a very
dedicated group of people
and I'm really proud of what
we achieved.

In case the previous discussion didn't make it clear, Isilon is highly committed to FreeBSD. A large part of our intent in moving to head was to allow us to contribute directly to FreeBSD without having to wait for our contributions to get into a stable branch or having to develop for both stable and head at the same time. We're also planning to upstream any of our internal modifications that make sense for mainline FreeBSD.

There are a bunch of areas we'd like to look at. One is the build process. The existing FreeBSD build is both awesome and flawed. On one level it's a magnificent thing that we can go into a directory, type 'make world', and have a complete OS pop out the other end. On the other hand, the way the build is structured makes it hard for Isilon to adapt it to what we do, which is to build world with a few extra parts, then build a bunch of ports, then build some more code that depends on the ports as well as the built world and then package the whole thing up. We'd love to work on a new build system that'd let us add our ports and port-dependent code in as first-level dependencies. We're looking at the bmake project that

Simon Gerraty's working on at Juniper and that may form the basis of something for us, but we'd love for the official FreeBSD build to become more flexible in this way. It'd also be really, really awesome to be able to build FreeBSD images on Linux or Mac OS X. Isilon doesn't really have a huge interest in building images for Raspberry Pi or BeagleBone Black or the like, but many people do, and allowing embedded developers to easily construct FreeBSD images from their desktop OS of choice would greatly increase the potential audience for FreeBSD on these platforms.

Another area we'd like to look at is the start-up process. We currently have our own internal service management daemon. It works, but it could be better. We can't use FreeBSD's stock init/rc code on its own as it doesn't do what we need. There are alternatives though, and we'd love to see something like launchd, possibly along with configd and notifyd, come in to FreeBSD or at least become available via ports. I think this is another area where FreeBSD could aggressively modernize. The old UNIX way of doing system startup is incredibly archaic these days, and by moving to these modern replacements, we get the benefit of a huge amount of thought from Apple and a set of service management daemons that give us all the benefits of something like systemd without running into the problematic areas of systemd itself.

We're also trying to forge connections with other FreeBSD-using companies so that we can coordinate resources on areas of mutual importance. This covers things like drivers, subsystem performance, VM improvements, and all of those things. We know that FreeBSD contributors are a limited resource, and if there are projects our engineers can work on that benefit us as well as other users of FreeBSD, then it's a definite win-win situation.

On the whole, FreeBSD's been very good to us. We want FreeBSD to continue to be good not only to us but to everyone it can. It's my fervent hope that we can make some awesome contributions to FreeBSD and that we keep using it for a long time to come. ●

BENNO RICE has been a FreeBSD committer since 2000 and is currently the manager of the OS team at EMC's Isilon division. His team looks after the FreeBSD code used in Isilon's OneFS. His interests include cooking, movies and upstreaming code.



FREE AND OPEN SOURCE SOFTWARE EXPO
AND TECHNOLOGY CONFERENCE

FOSSETCON

2015

Come out and participate in the Second Annual Fossetcon 2015
Florida's Only Free and Open Source Conference. With in
2 minutes of Downtown Disney and other great entertainment.

DAY 0

**FOOD, TRAINING,
WORKSHOPS AND CERTIFICATIONS**

DAY 1

**FOOD, KEYNOTES, EXPO HALL,
SPEAKER TRACKS**

DAY 2

**FOOD, KEYNOTES, EXPO HALL,
SPEAKER TRACKS**

BSD
Friendly

FREE FOOD,
TRAINING,
CERTIFICATIONS
AND GIVEAWAYS!!!

NOV 19 - NOV 21
Hilton Lake Buena Vista Orlando, FL

Fossetcon 2015: The Gateway To The Open Source Community

More info at
www.fossetcon.org

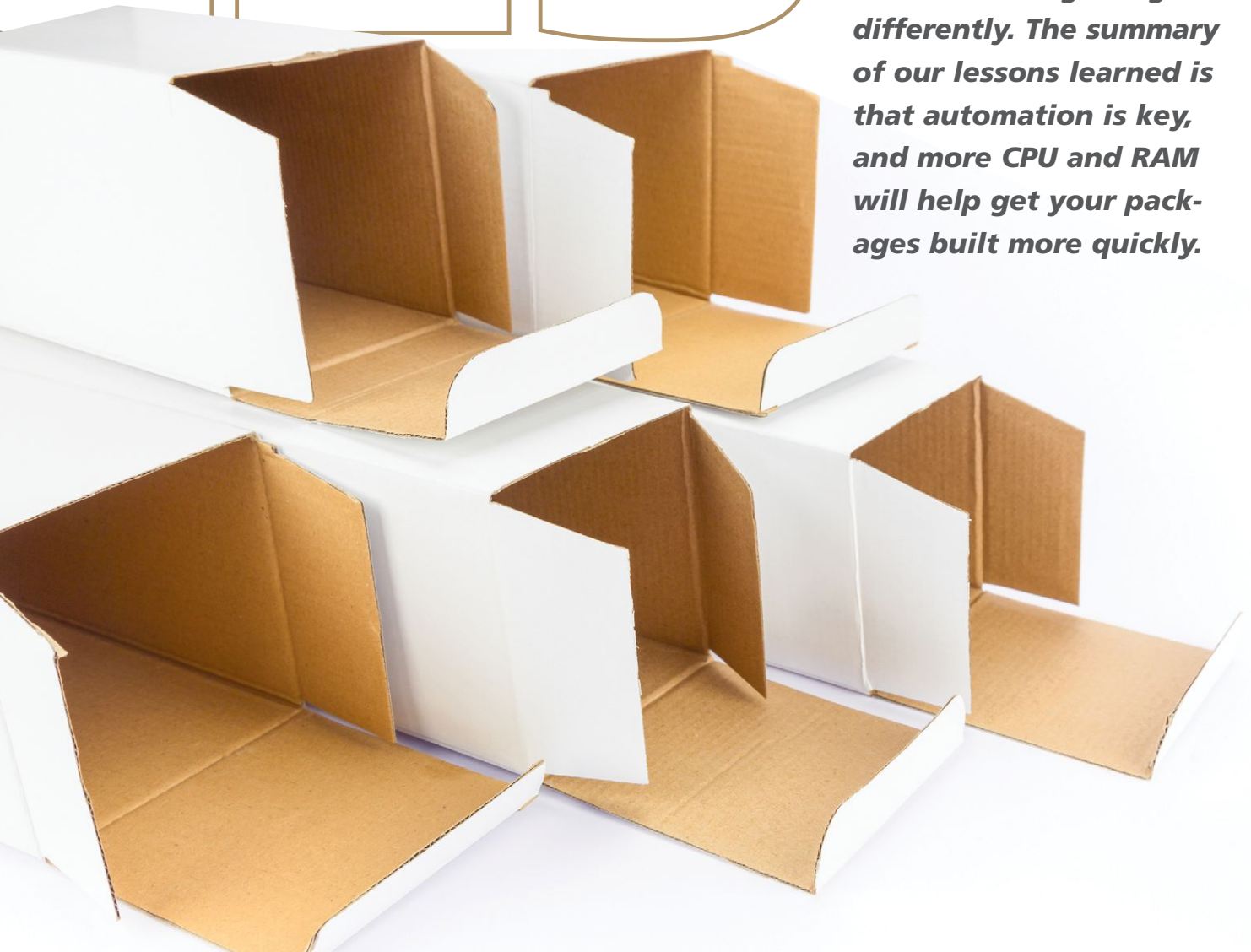
FreeBSD: Reflections on

PAC

PAC KAGES

by Bryan Drewery

I last wrote about how FreeBSD.org builds and provides packages in the March/April 2014 issue of this Journal. Since then, we have faced many challenges, made many changes, learned some lessons, and imagined a future of doing things differently. The summary of our lessons learned is that automation is key, and more CPU and RAM will help get your packages built more quickly.



Build Frequency

One of the biggest challenges we faced was the frequency of our builds. The frequency of providing packages is critical when there are security updates. For all of the OpenSSL vulnerabilities in the past year, the bash vulnerability, and some other high profile ones, we were unable to deliver packages in a timely manner. We often had the ports updated very quickly, a `freebsd-update Security Advisory` available out within a day, but the packages were not available for up to a week following. At the time, we were only using two servers to build 12 sets of official x86 packages—one server handling `pkg_install` packages, and another server handling experimental packages. The x86 builds on the two servers would typically consume four to five days of time per server, meaning that we could only deliver packages on a weekly basis. A full build of ports can take between 18 and 31 hours. An incremental build, only rebuilding what has changed, can take as little as one hour. The crux of the time problem comes down to how we do incremental builds, how we handle base system updates, and some long tail packages. Related to incremental builds is a fundamental problem with package failures. On package failures, we simply cannot provide a package for the run until it is fixed. (I will cover these issues in more detail later.) There has also been increasing demand for arm and mips packages and the work to provide those has matured to a point of us wanting to provide packages for them in the official repository.

The build time issues were too complex to solve overnight. In the meantime, we decided to seek out more hardware so we could meet the demands. Removing support for `pkg_install` allowed us to repurpose that server for the Pkg x86 builds, and we removed much of the need for the experimental server as well by making the packages it was building into the defaults. Over the past year the FreeBSD Foundation was generous enough to provide us with four additional servers consisting of 24 cores and 100GB+ of RAM each. We have dedicated six servers to the

x86 builds which means each server is only doing two builds per set. The average time of the builds now means we can typically complete an entire set in one day—due to the minimal time of 24 hours per set though we cannot guarantee daily builds yet. As of writing this, we are only scheduling the builds to run every other day to avoid getting systems out of sync. We need a build coordinator to handle each daily build so that once all are done, all can move to the next in unison. Currently each build is still independent, but is scheduled at the same time so they use the same revisions. A build coordinator is under development.

New Build Automation

Many new challenges were presented by having more servers and higher build frequency. A new goal was set to ensure that we could add any number of arbitrary new servers and just have things work after the initial setup, with the additional goal of allowing floating builds between each run. Our original build script was basically just a `poudriere` bulk from a `crontab`, along with an update to the jail and ports trees. A lot of things could and did go wrong that required manual intervention. Just adding more servers was a challenge as we needed to manually set up several items, despite using a puppet-like system already, and shuffle around packages, distfiles, and links. We also have another eight or so servers that do nothing but test and QA builds every day. Both the QA and Package builds benefit from the new automation.

The first step was to update the configuration management system to properly track all required files and install them such that each system was identical. Combining this with the arm/mips builds using QEMU means that we do not need to do manual setup for those systems. The config management and build script both now properly set up everything needed, from scratch, for the builds it will handle. In terms of emulation, we also want to ensure that if we deploy our builder to a native mips system, that it does not use QEMU. Both `Poudriere` and our build scripts will handle this situation properly and not use QEMU by respecting `sysctl kern.supported_archs` and only using QEMU if the arch-to-build is not in the list.

On every build, the script would update the

jails to the latest version of the release they are tracking. For head jails this means building from `src`. If the build were to fail, then no head packages would be provided until the next run, which could be up to a week. With daily builds we wouldn't want to have packages one day and not the next. It would hurt the user experience too much. Now the script will roll back to the previous jail version if the build fails.

For the idea of floating builds, we want to allow a build to happen on serverA today and serverB tomorrow. The reasoning being that it gives us less configuration to manage, more flexibility to move servers around or add new servers, and allows the build coordinator more flexibility. This was useful immediately as we ended up needing to ship four of the servers in the pool across the country shortly after deploying the new system. To allow floating builds, we need to sync distfiles and the previous package set into the system before attempting a build so that no time is wasted refetching distfiles or rebuilding packages that do not need to be rebuilt using the normal incremental algorithm. Previous to this rework, each build server was fetching distfiles directly from the Internet, even for ports that had not changed in years. In the past year we set up a new distfile cache on FreeBSD.org. In a sense this is our site cache. So now the builds will use this cache first via `MASTER_SITE_FREEBSD=yes`. For your own caching you could use `MASTER_SITE_OVERRIDE=yoursite.com` to try fetching from first, or just use a caching proxy. This simplified the package build distfile fetching and allowed parallel seeding of distfiles into new systems in a quick manner. The other seeding need was the package set. Now the build will always attempt to seed in packages from our local mirrors so it can benefit from an incremental rebuild.

If somehow a patch slipped into the ports SVN checkout, it could impact the resulting packages in unintended ways. This was never a problem for package builds, but was a problem for our test builds, since we are often manually applying patches to test for other people. With a pool of servers, it is possible that a package build occurs on a system that was previously used to test a patch. Now the script ensures everything is reverted.

The ports quarterly branch's structure is such that the path used for checkout changes every quarter. We will copy `/head` to something like `/branches/2015Q2` and then cherry-pick commits from head to the 2015Q2 branch as appro-

priate, for security and other critical fixes. However, in the next quarter, we would then copy head to 2015Q3 and leave 2015Q2 abandoned. It is not trivial to just merge all of head such that all of it replaces the files in 2015Q2 to be able to retain the branch. Often the changes in the quarterly branch can be different from head. A direct merge won't work. We could replace all of the files from head and commit that, but we opted for the simpler approach for now. Because of the churn of the "current" quarterly branch location, we would also need to manually svn switch the package builder checkouts as well. Now that is automated. (I will discuss more about quarterly packages later.)

Due to the increased frequency of builds and the lack of being able to retain previously-passing packages for now-failing packages, we needed to ensure that accidents did not wipe out packages for a day. If someone were to update `gettext` and make a mistake or do poor testing, then it would previously have caused all packages that depend on `gettext`, indirectly or directly, to not be published and not be provided from the repository until it was fixed and a new build done. We have put in place a simple short-circuit of the publishing if too many packages fail. Currently that number is 1,500 packages, which is 6% of the 24,000 ports in the tree. We discussed a list of blockers, and are still considering it, but did not implement it, as it could easily too often block security updates. Even recently, we had a broken gnome package for a long period of time. Should gnome being broken prevent a critical openssl update from going out? No, but it does hurt user experience greatly when popular packages are broken. Dealing with that is a long-term issue that is discussed more later.

Finally, with floating builds we can easily confuse users who are used to looking at the same server for package build information. We have a system in beta at <http://pkg-status.FreeBSD.org> that is tracking all of the builds and displaying these in aggregation and allows searching of specific packages to see results from the previous runs. The code for the site is available at <https://github.com/bdrewery/pkg-status.freebsd.org>. Due to security concerns, we keep the on-build web system from Poudriere 100% static without any server-side daemon or code execution per web request. The `pkg-status` site gives us far more flexibility by allowing us to provide a real web API for querying the status of builds for scripts. This site will continue to improve over time. Eventually we will likely disable direct

access to the builders and have `pkg-status` keep the build logs for all packages from the builders. This will greatly simplify users' looking for build results and increases security by disallowing access to our private builders.

Incremental Rebuild Issues

The way we currently build packages has risen from historical need and ease. Now that we have reached a point of being able to provide packages, we need to do more research and rethink how we do things. Some of the problems are discussed in detail next. Feedback and help are needed.

Most of the incremental and failed build issues we have today are due to what I will call our Monolithic ports and lack of modern packaging ecosystem. Ports only reliably works as an entire snapshot. The problem with failed builds is easiest to understand with a real example. Consider if Chromium fails to build. Let's say it is only being rebuilt because a dependency changed—Chromium was not actually updated, but it fails to build due to some other reason. We have the old package, so we could in theory just restore that and supply it in the repository. However, what about the dependencies? If any of the library dependencies was updated and was not ABI compatible, then the restored package would be looking for an older library that is no longer provided. So we would also have to provide the older library dependency package. How can we provide two versions of the same package for the library dependency when they conflict on all files except the library? In other packaging ecosystems there is an idea of "alternatives" (not to be confused with the specific Debian implementation). This is a critical feature that we currently lack. Alternatives would allow multiple versions of the same package to be installed into something such as `/usr/local/opt/PKGNAME-PKGVERSION` and allow the "default" to be user-selectable and installed via symlinks in `/usr/local`. This is the only way to automatically provide two normally-conflicting packages into the same system.

A lot of people are quick to say that we are relearning all the mistakes of other systems, but don't see that we lack critical pieces to make things work properly. If we had an alternatives system, then we could, in theory, provide the

previous library dependency here and still provide the old Chromium package. However, in some cases, we would need to modify the restored package to change dependencies, and worse, we would need to do this same restoration recursively and for all other failed builds as well. It easily balloons out of control. It's hard to test these ideas and to see what the real world results would be without the ability to have multiple versions of the same package/origin installed concurrently. Note that we can install packages concurrently, but they must be built to

“THE
BIGGEST PROBLEM SEEN IS
A LACK OF COMMITMENT FROM
COMMITTERS. WE NEED HELP HERE, TOO”



not conflict. Alternatives is a desperately needed system. An alternative approach to this entire problem is with a PBI-like package system. In PBI all the dependencies are built right into the resulting package. So for Chromium we really could just provide the previous package as it would have its dependencies bundled. I am not that familiar with PBIs, but my understanding is that it is smart in deduping redundant files on install, but would still have these mega packages to be downloaded and stored. For Windows, OSX, Android, iOS, etc., packages, it is clearly bundling dependencies and, for the most part, managing fine. Perhaps PBIs and how we do packages needs to be reconsidered.

The other problem with incremental rebuilds is that we simply don't know what needs to be rebuilt and when. We used to trust ports committers to "chase" a dependency update by bumping the `PORTREVISION` for all ports depending on the port they updated. It is not safe to assume human intervention will do the right thing though. As such, if a dependency is updated, then Poudriere will force-rebuild anything that depended on it. If Poudriere had a way to

know that a dependency has a real build-time or run-time impact on the build of other ports, then it could avoid forcing a rebuild. The usual case of the need for this is on toolchain dependency that can alter the output of a package's binaries and libraries. Obviously if a library's major version is incremented, then a rebuild is needed. What if it is not bumped though? The upstream maintainer's actions on the library version are not always correct with ABI stability. There are tools called ABI compliance checkers specifically for checking whether or not a library has retained its ABI. We could probably use these to improve the incremental rebuild of Poudriere. Outside of ABI compliance though there is a lot of uncertainty. Will an update to bash need to be considered a "toolchain" update? How will we identify all "toolchain" ports easily without making a mistake? It is not reasonable to require committers to chase dependencies. An update to the ports compiler or a highly used dependency would require touching up to 24,000 ports. When the ports GCC is updated, it has never chased other ports. Poudriere does though and ensures that the new GCC is used. Of course all this greatly hurts reproducibility as well. One package's version may be different today and tomorrow depending on the rebuilds done to it. This is against our goals. We need help in researching the need for incremental rebuilds and integrating an ABI compliance checker. Lacking any of this, Poudriere must be aggressive with its rebuilds, which wastes a lot of time.

I mentioned earlier a problem of long tail packages. During a typical package build most of the packages will complete around 12 hours in, but a handful will continue to build from hour 12 to hour 16+. These packages, such as OpenOffice, PyPy, and Paraview, to name a few, are demanded by users specifically because they take so long to build. Often we do rebuild them when not needed. However, when we have 24

cores at our disposal and Poudriere is only building two ports for five hours, there are a lot of resources being wasted. Poudriere limits each build to 1 CPU, via the ports

MAKE_JOBS framework. When these long tail package builds start, the system is usually at full load.

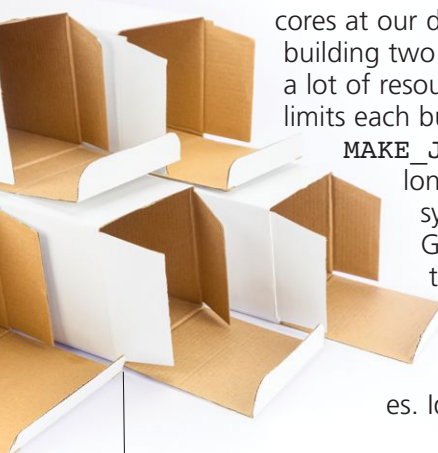
Giving many cores to each of these would help speed up the overall build, but will overload the system quite fast before everything else finishes. Ideally, Poudriere could dynami-

cally assign cores to `jails / make` and allow the builds to scale up or down. We could probably play with `cpusets` a bit, but it would require lying to the build of these ports and telling them they can use 24 cores when really the `cpuset` we provide is only one core. We have not experimented with this, but it seems like it might be counter-productive. Another option is to have Poudriere be smarter and recognize the situation and spawn the next set to build. That's not simple currently as Poudriere is `jail-release-arch-portstree-set` specific. If it had a master queue handler at which we could pass arbitrary port builds, then this situation would be easier to handle. That would require keeping some worker jails spawned up and idle. This is reasonable, but has not been implemented.

The final issue with package rebuilds is when the base system is updated. This happens hourly for head for which it is reasonable to rebuild packages. The problem is with SA (Security Advisories) and EN (Errata Notes). Often these will change something in the base system that really has no impact on packages. Sometimes it does though. This is very similar to the ABI issue, but in a grander way. We used to consider the `__FreeBSD_version` number in `/usr/include/sys/param.h` as indicating if the "ABI" of the OS was changed. However, it is often missed and typically never modified for SA/EN. So for package building, the safest solution was to just always rebuild all packages when the jail was updated. I do have a plan for resolving this and plan to implement it directly into Poudriere. It is discussed on the `freebsd-arch@` mailing list in more detail at <https://lists.freebsd.org/pipermail/freebsd-arch/2015-April/017025.html>. The summary is that we will track files in the OS that typically do impact packages, such as headers and the toolchain. Only if any of these files are updated will we force a rebuild of packages. This same logic will be extended to the head builds as well, as it is not always necessary to rebuild for that branch as often it is for just a kernel or documentation update.

Stable Packages

The final topic is the idea of a stable branch. After we increased the build frequency of the packages, some of the first feedback we received was that it was being updated too often now. There is no good balance here. We must be able



to release security updates quickly, but we also do not want to churn user's systems too often. The quarterly branch exists for this reason and perhaps should be the default for packages. However, it is a half-measure of stable packages. We update it 100% from the head branch every three months. So you will only really get a month or two of stability and then a burst of changes and instability. The branch is not created from any stable point; it is simply created when the time comes. In my opinion, it would be better if we had people dedicated to maintaining long-term stable branches just as we do for the base system. For the base system, we have the head branch locked for a period of time and a team ensures it is stable through time, testing, and review. They then copy it out to a new /stable branch for the release. Never in the base system do we suddenly make /stable/10 into /head like we effectively do with the ports quarterly branch. We make users decide when to update for the base system. Ports used to do these periods of lockdowns, but only right before releases. Somehow in the time after we moved to SVN portmgr, the group responsible for ports—of which I am a member—has taken on the wrong approach to branching, and there has been little motivation to resolve it. Currently we provide only one set of packages for the quarterly branch that churns very fast and will not deliver the expectation that users want. A stable branch would need a balance between how we do packages now (freshness) and stability. Most people, including me, are against a stable branch in which you have five-year-old software like some other OS distributions have. I do think it would be beneficial to follow a similar workflow as the base system and do actual port "releases." This involves a ton of work and careful consideration of ABI stability and manually backporting patches. The biggest problem seen is a lack of commitment from committers. We need help here, too. •

BRYAN DREWERY has managed shared hosting services with FreeBSD since 2004. He joined the project in 2012 as a Ports committer, is a member of Portmgr and has recently become a Src committer. He is the current upstream maintainer of Portupgrade, Portmaster, and a developer on Pkg and Poudriere. In Portmgr, he helps with the Ports framework, managing the package build systems, package building and testing ports patches on them.



atlantic.net

FAST SSD CLOUD HOSTING

Up in 30 Seconds
24/7 Support
Per Second Billing



One-Click Apps





LAMP

LEMP

Starting at
\$4.97
per month



Inside The FreeBSD Foundation

/ by **Deb Goodkin**

These past few months just flew by! When you have a small staff, everyone keeps busy reaching out and helping the FreeBSD community. We couldn't accomplish what we do without a passionate, committed team of people who go above and beyond what's expected every day. I'm proud of my team and am here to shed a light on what we've done the past couple of months to support and grow FreeBSD.

The FreeBSD Foundation's sole purpose is to support the FreeBSD Project and community worldwide. We do this in many ways, such as fund development projects, purchase equipment to improve FreeBSD infrastructure, sponsor conferences and FreeBSD developer travel expenses, promote FreeBSD, and provide FreeBSD education opportunities.

Bringing FreeBSD People Together

We participated in two conferences in the last couple of months. We were proud to be a Platinum sponsor for BSDCan 2015 in June in Ottawa, Ontario, Canada, as well as the primary sponsor of the Developer and Vendor Summits that preceded BSDCan. It's hard to believe this was our 10th year sponsoring this conference!

This is the one event that all Foundation board members attend. When I first joined the Foundation, 10 years ago, we started having our annual board meeting during the conference. It was more of a formality, because we had to hold our elections every year. We'd pick a night to meet for one to two hours to hold our elections and talk about FreeBSD development work. We've grown so much that our annual board meeting has become a formal all-day meeting preceding the summits. Although it makes for a long week, it is extremely productive for our team to work face-to-face on strategic planning, long-term goals, project roadmaps, fundraising, FreeBSD advocacy, and holding our elections. We also elected the current officers

and directors (here), welcoming Benedict Reuschling as a new director.

Following our board meeting (<https://www.freebsd.foundation.org/board>), we all attended the Developer and Vendor Summits. Ed Maste, our project development director, ran the Vendor Summit. This is an opportunity for developers and vendors to share their project direction and goals, and collaborate on projects of broad or mutual interest. We spent half a day discussing a list of technologies that companies would like to develop and/or upstream to FreeBSD.

During the Developer Summit, our marketing director, Anne Dickson, presented "*FreeBSD Advocacy: How You Can Help Spread the Word*". We had a great turnout for this, and it ignited a lively discussion on how to promote FreeBSD.

When Dan Langille opened the 12th annual BSDCan conference Friday morning, the room was filled with this magnificent amount of energy. There were so many attendees that an over-fill room was set up.

During the opening session, I gave a short presentation on the Foundation and what we do to help FreeBSD. On my first slide I had a comment from Dan's Facebook page that said, "So many smart people in one room. This is why I attend BSD conferences." It's true—some brilliant people attend these conferences. One first-time BSD conference attendee and travel grant recipient said to me, "the BSD community is not only incredibly smart, but also just as nice, and they made me feel right at home."

During the morning presentation I highlighted two things that generated a lot of excitement from the audience. One was our women-in-tech initiative, where we are working on recruiting more women to FreeBSD. The second was the course curriculum work we are sponsoring, as well as our ideas of bringing FreeBSD education to college, high school, and middle school students. I can't tell you how many people approached us afterwards, wanting to know

how they can help or to connect us with people at their universities.

Foundation team members gave talks, attended talks, participated in doc sprints, worked on efforts to improve FreeBSD, worked at our booth, and spent time talking to our constituents about areas where we can help with FreeBSD.

Including those listed above, the Foundation board and staff members gave the following presentations:

Anne Dickison: *"FreeBSD Advocacy: How you can spread the word"*

Kirk McKusick: *"An Introduction to the Implementation of ZFS"*

George Neville-Neil: *"Measure Twice, Code Once"* and *"Cambridge L41: Teaching Advanced Operating Systems with FreeBSD"*

Ed Maste (who also ran the Vendor Summit): *"The LLDB Debugger in FreeBSD"*

When I reflected back on the conference and read trip reports from the 12 FreeBSD contributors whom we sponsored, I thought about how the presentations inspired thoughtful discussions, not only in the sessions, but long afterwards. The face-to-face opportunities for FreeBSD contributors have a profound effect on the Project. Not only is a lot of work started, and accomplished, at these conferences, but people work together on projects long after the end of the conference. It's an opportunity to share knowledge and to learn, whether you are new to the Project or have many years under your belt.

The other conference that Anne Dickison and I attended was OSCON, where we spent two-plus days talking to people about FreeBSD and the Foundation. Not only were people interested in FreeBSD, but many were interested in offering FreeBSD workshops at their meetings, teaching FreeBSD in their operating systems classes at universities and community colleges, and using FreeBSD in computer classes offered to disadvantaged kids. We also spoke with people looking to try out FreeBSD for the first time, or going back to it again after many years away. It was very satisfying to see people become genuinely enthusiastic about trying it after hearing all that FreeBSD has to offer. Overall, OSCON 2015 turned out to be a valuable event for FreeBSD and the Foundation.

Events like OSCON help us in our mission to bring a FreeBSD presence to non-BSD-focused conferences, like open source, women in tech, systems administrators, and data storage, to promote FreeBSD, to educate people about the operating system, and to recruit more people to the project. The Foundation is helping to get FreeBSD contributors to work at FreeBSD booths and give presentations at conferences like OSCON, FOSSDEM,

womENCourage, and more. We provide travel grants for FreeBSD contributors to attend conferences, and provide literature to hand out. As we continue to identify FreeBSD people in different regions who are interested in promoting FreeBSD at conferences in their areas, we recognize even more how invaluable and cost effective this procedure is for us and the Project.

Following are the upcoming conferences that we've committed to sponsoring:

vBSDCon, SNIA Storage Developer Conference, womENCourage 2015, EuroBSDCon 2015, Grace Hopper conference, BSDCon Brasil, Cambridge Developer Summit, Bay Area Developer Summit, USENIX LISA, and OpenZFS.

Funding Project Development Efforts

Our biggest ongoing project is the FreeBSD/arm64 port, and a lot of progress has been made over the last couple of months. Recent improvements include ACPI, SMP enhancements, support for DTrace and hwpmc performance profiling. As community interest in the port has been increasing, FreeBSD developers are working on supporting the platform in different ways. Over 14,000 third-party packages successfully build now, and changes are in progress to enable another 5,000 or more packages to build.

FreeBSD is now running on 48 real ARMv8 cores! We have four dedicated staff members who can focus full-time on code fixes and improvements, new features and functionality, and release engineering and cluster administration. Here's some of the work our staff completed over the past few months.

- Edward Napierała continued work on the reroot project, which allows the system to boot with an initial root file system, perform some configuration and setup tasks, and then replace the temporary root with the real one.
- Konstantin Belousov continued work on the DMA remap (DMAR) project, and also committed fixes to a number of kernel subsystems. Konstantin is also working on improving atomic operation support, with a goal of improved correctness and performance.
- Foundation employee Glen Barber along with the Release Engineering Team worked on the FreeBSD 10.2 RELEASE, starting with the code freeze at the beginning of the month and keeping release notes up to date throughout the process. By the time you read this article, FreeBSD 10.2 should be available.

Glen continued adding support for building virtual machine images for a number of targets. The Release Engineering Team supports three thirdparty

Inside The FreeBSD Foundation

(also known as “cloud”) providers—Amazon EC2, Microsoft Azure, with support for Google Compute Engine expected in the final 10.2 RELEASE as well. Glen also performed general maintenance and updates to the FreeBSD.org infrastructure with the FreeBSD Cluster Admins team.

- Ed Maste continued working on a number of improvements to the FreeBSD software development tool chain. He worked with other developers in the ELF Tool Chain project to identify and fix a number of bugs that affected FreeBSD ports builds, and imported updated versions of the tools into FreeBSD. These tools are now fully functional, and the infrastructure to build the old versions of GNU Binutils tools has been removed. Ed also continued with the LLDB port for FreeBSD as well as the arm64 porting project, and collaborated with FreeBSD developers on initial investigation into the RISC-V instruction set architecture from Berkeley.

Educating the World

It is a bold statement, but that is our initiative. We are working on classes and workshops to teach FreeBSD from middle school to graduate level students. Robert Watson and George Neville-Neil are continuing their work on a college level class based on their book *The Design and Implementation of the FreeBSD Operating System, 2nd Edition*, by Kirk McKusick, George Neville-Neil, and Robert Watson. The course is based on operating system tracing and experimentation on the BeagleBone Black platform. This course was successfully taught by Robert as a graduate level class at the University of Cambridge last school year, and will be taught again this fall.

The Foundation is sponsoring George to spend the month of August in Cambridge to work with Robert on refining and expanding open-source teaching material, that will be available online later this summer. George is also working on introducing this curriculum into a few US universities.

This fall, we are going to offer our first middle school class, here in Boulder Colorado. Justin Gibbs is currently working on the curriculum and will be using the BeagleBone Black for the teaching platform. We're thrilled that local FreeBSD contributors have offered to assist as guest teachers. We are also working on a three-hour Introduction to FreeBSD workshop that we can start offering in January. I'll have more to report on our progress in the next Journal issue.



Giving FreeBSD a Voice

We are determined to make more people aware of FreeBSD and encourage people to try it. We are doing this by creating more informative literature that we can hand out at conferences. We've made these brochures available online, so FreeBSD advocates around the world can translate them into their languages and provide at conferences where they are promoting FreeBSD. We know people love showing their support for FreeBSD with stickers. So, we've created cool new ones that will stand out and make a statement. We've been reaching out to companies to provide FreeBSD testimonials, with new ones being published on our website and in our newsletters monthly. We were thrilled to receive one from Microsoft last month. We're helping to get FreeBSD presence and presentations at more conferences like OSCON and SNIA SDC.

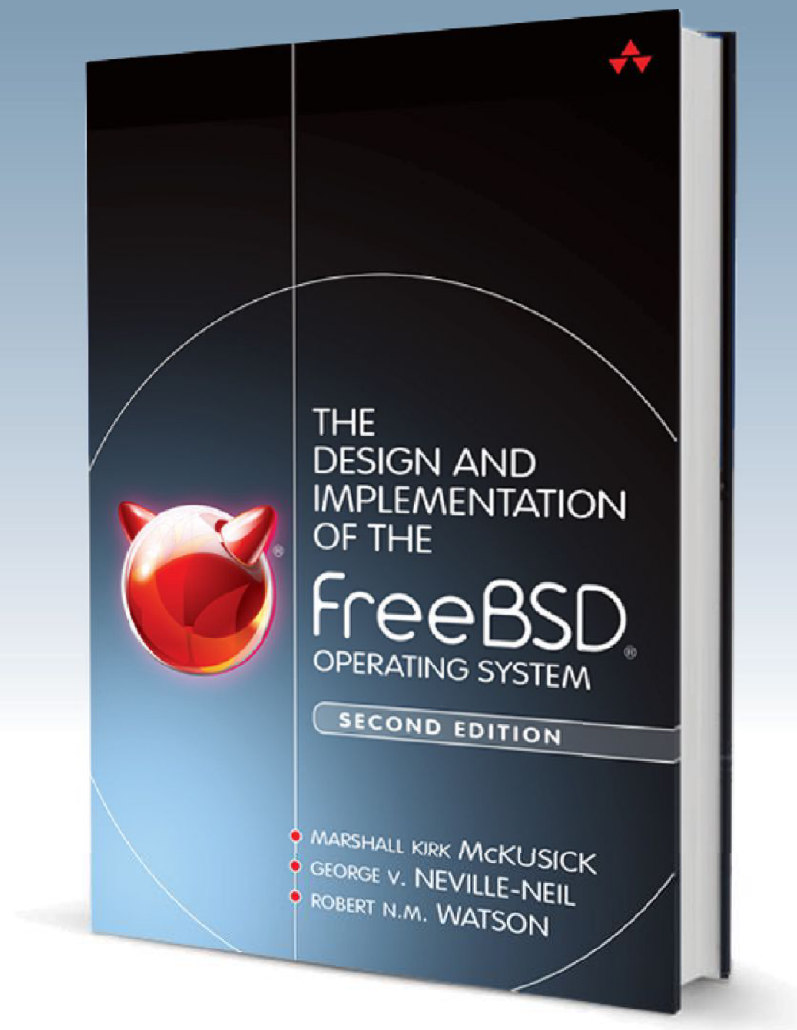
I'm pleased to report the *FreeBSD Journal* has over 9,800 subscribers! We are extremely proud of the publication and the authors who give their time freely by contributing their informative articles. This has been a great source of helpful and useful information for all Unix-like operating systems. As the number of subscribers and advertisers increases, it will help us reach our goal of a self-funding publication. If you work for a company that uses FreeBSD, consider putting an ad in the *Journal* to advertise your product, services, or job opportunities.

To find out more information about the Foundation and what it does, please go to www.freebsd.foundation.org. You can also subscribe to the Foundation's monthly e-newsletter by signing up at the bottom of our website.

Finally, as you may know, the Foundation is entirely supported by donations from people like you. Please consider helping us continue the work that we are doing by making a donation and encouraging your colleagues and companies to do the same. Go to <https://www.freebsd.foundation.org/> and make a donation today! We can't do it without you. •

Deb Goodkin is the Executive Director of the FreeBSD Foundation. She's thrilled to be in her 11th year at the Foundation and is proud of her hard-working and dedicated team. She spent over 20 years in the data storage industry in engineering development, technical sales, and technical marketing. When not working, you'll find her on her road or mountain bike, running, hiking with her dogs, skiing the slopes of Colorado, or reading a good book.

NEW EDITION – Now Available!



The most complete, authoritative technical guide to the FreeBSD kernel's internal structure has now been extensively updated to cover all major improvements between Versions 5 and 11.



SAVE 35%

When you order from informit.com/freebsd
Use discount code **FREEBSD35** during checkout

EBOOK FORMATS INCLUDE EPUB, MOBI, AND PDF – ALL FOR ONE PRICE • FREE SHIPPING WITHIN THE U.S.

Terms & Conditions: Discount code FREEBSD35 is applied to list price of Design and Implementation of FreeBSD, Second Edition print or eBook and cannot be combined with any other offers. Offer is only good at informit.com.

informit.com
the trusted technology learning source


Addison
Wesley

conference **REPORT**

by Shonali Balakrishna

BSDCan 2015 (www.BSDCan.org/2015/)

One slide at the opening session of the BSDCan conference read, “So many smart people in one room. This is why I attend BSD conferences.” I smiled to myself and began to soak up the excitement of my first BSD conference. After nine talks and multiple conversations with the aforementioned smart people, I am thoroughly convinced! Not only does a BSD conference have way too many very smart people in one room, they are also some of the nicest. You hear about the great work done by the FreeBSD community all the time, but not enough good things can be said of the incredible FreeBSD community and the inspiration it provides.



I arrived the first day of the scheduled talks after a grueling finals week at graduate school. I was definitely sleep deprived, but the excitement and the energy at the conference kept me wide awake. I found each of the nine talks I attended very interesting. At the plenary, it was incredible listening to THE Stephen Bourne talk

(<https://www.youtube.com/watch?v=2kEJoWfobpA>) about the creation of the Bourne shell and his many insights and inside stories. I loved the part where he explained how he convinced Dennis Ritchie to inculcate the ‘void’ type into C and enjoyed his candor about the debugging of shell scripts.

I especially enjoyed the following talks: FreeBSD Operations at Limelight Networks—content delivery networks (CDN) interest me, having had a research component on it at graduate school this year, and to see it in the operations perspective, with their usage of FreeBSD, was very interesting. They deploy their own backbone and use FreeBSD on their backbone. For a big CDN with many data centers, large-scale installations of FreeBSD on the edge is an important factor. Such an operations workload requires fluidity in software and configuration changes—and only a handful of people are involved in the design and operations at Limelight Networks when compared to the usual model. The use of FreeBSD allows both, as you are pulled into the source tree and can get involved in the operations of the system. While deploying FreeBSD, their strategies involve

upstreaming everything, using ports, and creating a src team. Tools used by Limelight were discussed—Zabbix for monitoring to ensure API-driven configuration management, monitoring in test, dev and QA to form an efficient feedback loop, OpenTSDB for storing time series metrics, SaltStack for configuration management using a declarative style where the system takes action based on policy using an orchestration bus, and Vagrant for pushing FreeBSD on the edge servers.

CloudABI by Ed Schouten is a Unix application binary interface that provides capability-based security. It extends the concept of Capsicum by providing a more compact representation with only about 60 system calls. This can be applied to cloud-computing environments instead of using full system virtualization or virtualized namespaces. CloudABI can be used in a far simpler way and with better performance without complicated configurations while providing security by controlling access to sockets, files, and other resources. It provides a whole new way of creating environments by building on capability-based security to provide isolation and resource access control at the process level rather than complete isolation of environments to provide the level of security required.

An Introduction to the Implementation of ZFS by Kirk McKusick was an interesting talk for more reasons than the fact that it was Kirk McKusick himself talking about ZFS. There was also a fire alarm going off in the middle of the talk, followed by the arrival of fire trucks! I loved the talk for the clarity with which Dr. McKusick explained the ZFS block structure and checkpointing, and he

PHOTO BY OLLIVER ROBERT

made it easy to understand the rather complex concepts like freeing of file system and snapshot blocks in ZFS. The closing session hosted by Dan Langille was a lot of fun, and again the awesomeness of the BSD community was on shining display. I attended the closing party at the nearby Lowertown Brewery, which was enjoyable and a good opportunity for some networking as well.

It was great to finally meet Gavin Atkinson, who was the administrator for all of us GSOC students at FreeBSD last summer. I also got to converse with Diane Bruce, Ed Schouten, Deb Goodkin, Justin Gibbs, and David Chisnall, among others, and every conversation was refreshingly unique in the perspectives it provided on the work done at FreeBSD, specific projects in FreeBSD, technology in general, the codebase, ways to contribute to FreeBSD, history, music, and a myriad of other subjects.

In terms of the work I'd like to do for FreeBSD in the future, besides wrapping up the testing of the BSNMP IPv6 project, I learned more about other projects like further IPv6 support (userland cleanup, unification of ping and ping6, unification of traceroute and traceroute6), improvements to the Bluetooth module, 802.11 improvements, and

Space Communication Protocol.

Ottawa was an incredible city to visit—steeped in history and culture—and I soaked in the museums, art galleries, and beautiful hiking trails it provided. Even though I arrived there mentally exhausted, I returned home completely excited and inspired, with many plans for future contributions to FreeBSD, and so very grateful to the FreeBSD Foundation for having provided me with this opportunity to attend my first (of many more, hopefully) BSDCan conference. ●

SHONALI BALAKRISHNA was a 2014 Google Summer of Code student for The FreeBSD Project and contributed to the BSNMP project, where she worked on adding IPv6 support to BSNMP. She is currently a Master's student at the University of California, Irvine, majoring in Networked Systems. Shonali holds a Bachelor's degree in Telecommunication Engineering, and her Bachelor's research and thesis led to two research publications. Her current research interests are at the intersection of Distributed Systems and Machine Learning, and she also hopes to continue contributing code to FreeBSD.

Thank you!

The FreeBSD Foundation would like to acknowledge the following companies for their continued support of the Project. Because of generous donations such as these we are able to continue moving the Project forward.



The
FreeBSD
FOUNDATION

Are you a fan of FreeBSD? Help us give back to the Project and donate today!
freebsdfoundation.org/donate/

Iridium



Platinum



Gold



Silver



Please check out the full list of generous community investors at freebsdfoundation.org/donate/sponsors



We're excited to announce that March 15, 2015 marks the **15th Anniversary** of the FreeBSD Foundation!

You've helped up accomplish so much during the last 15 years and we look forward to continuing that progress through out the rest of 2015. The areas we'd like to invest in include the following:

- Funding Projects to Advance FreeBSD
- Increasing Our FreeBSD Advocacy and Marketing Efforts
- Providing Additional Conference Resources and Travel Grants
- Continued Development of the FreeBSD Journal
- Protecting FreeBSD IP and Providing Legal Support to the Project
- Purchasing Hardware to Build and Improve FreeBSD Project Infrastructure
- And More!

Thank you for all of your continued support. We can't do it without you!



Support FreeBSD®

Donate to the Foundation!

FreeBSD is internationally recognized as an innovative leader in providing a high-performance, secure, and stable operating system. Our mission is to continue and increase our support and funding to keep FreeBSD at the forefront of operating system technology.

For 15 years, the FreeBSD Foundation has been proudly supporting the FreeBSD Project and community thanks to people like you. We are incredibly grateful for all the support we receive from you and so many individuals and organizations that value FreeBSD.

Make a gift to support our work in 2015. Help us continue and increase our support of the FreeBSD Project and community worldwide!

Making a donation is quick and easy.
Go to freebsdfoundation.org/donate



The
FreeBSD
FOUNDATION
freebsdfoundation.org

PORTSreport

by Frederic Culot

During the May–June period, the activity on ports was very high, with an increase of more than 20% of the changes applied to the ports tree compared to the March–April period! A lot of improvements were also brought to the ports framework as described hereafter.

NEW PORTS COMMITTERS AND SAFEKEEPING

In the May–June period, one new committer was granted a ports commit bit: Bernard Spil, who is mentored by vsevolord@ and koobs@. Some commit bits were also taken in for safekeeping, either on a committer's request (this was the case for sahil@), or following a period of inactivity of more than 18 months (this was the case for clsung@, dhn@, obrien@, and tmseck@).

IMPORTANT PORTS UPDATES

Many exp-runs were performed by Antoine@ (23 in total), to check whether major ports updates are safe or not. Among those important updates, we can mention the following highlights:

- default perl version set to 5.20
- python (2.7 branch) updated to 2.7.10
- cmake updated to 3.2.3
- bison updated to 3.0.4
- ffmpeg updated to 2.7.1

As usual, please read the /usr/ports/UPDATING file carefully before updating your ports, as manual steps may be involved.

PORTS FRAMEWORK UPDATES

Some changes were applied to the ports tree infrastructure that might not be readily noticeable by end-users. This involved removing deprecated functionalities (such as the

removal of the NEED_ROOT macro, since all packages could now be built as a regular user). It also concerned improving some functionalities, such as the way dependencies are checked and installed. These are important steps to prepare the ports tree for flavors and subpackages, so stay tuned as these important new features are expected soon! More generally, it is interesting to check

the commit logs for critical infrastructure components, such as for the files found in /usr/ports/Mk, as that often gives a good overview of what's coming next. Those commit logs are available online, such as for [bsd.port.mk \(https://svnweb.freebsd.org/ports/head/Mk/bsd.port.mk?view=log\)](https://svnweb.freebsd.org/ports/head/Mk/bsd.port.mk?view=log).

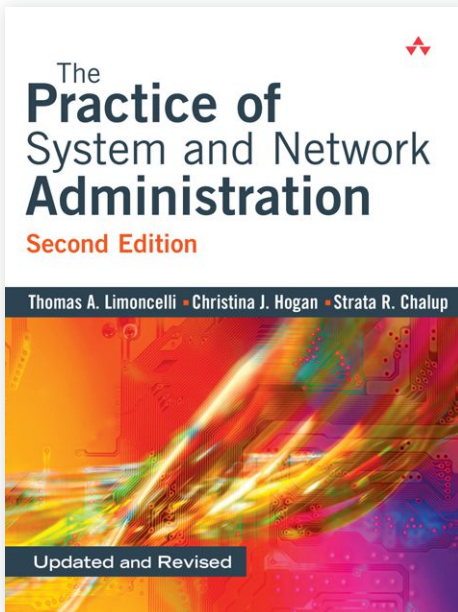
STATS

As always, a few statistics to finish: for the two-month period from May to June, 5,595 commits were applied to the ports tree, which is an increase of more than 20% compared to the last period. On the bug reports front, 1,253 PRs were closed, which is just a slight decrease compared to the March–April period. Let's thank all our contributors for their hard work, and let's hope that many more new developers join our ranks in the months to come. •

FREDERIC CULOT has worked in the IT industry for the past 10 years. During his spare time he studies business and management and just completed an MBA. Frederic joined FreeBSD in 2010 as a ports committer, and since then has made around 2,000 commits, mentored six new committers, and now assumes the responsibilities of portmgr-secretary.

BOOKreview

by Greg Lehey



The Practice of System and Network Administration

by Thomas A. Limoncelli, Christina J. Hogan and Strata R. Chalup

- Pearson/Addison-Wesley Professional • 1,056 pages
- ISBN: 9780321492661
- Print List Price: \$59.99*
- eBook List Price (includes EPUB, MOBI, and PDF): \$47.99*

*For FreeBSD Journal readers: To obtain a 35% discount, visit this publisher page for both of Limoncelli's books (<http://informit.com/tposa>) and apply this code at Checkout: **TPOSA35**

Publisher's note: The 3rd Ed. is scheduled to release in spring 2016.

What do you look for in a book on systems administration? My benchmark for such books is *Unix® and Linux® System Administration Handbook*, by Evi Nemeth, Garth Snyder, Trent R. Hein, and Ben Whaley (which I abbreviate to Nemeth below). *The Practice of System and Network Administration*, by Thomas A. Limoncelli, Christina J. Hogan, and Strata R. Chalup (Limoncelli for short), is very different. Why?

To answer that question, it's worth stepping back a bit and looking at what you want from a book about system administration. Although I greatly admire the Nemeth book, I have hardly ever used it. I have many books, and when I'm looking for information, I may choose one of them, or increasingly find information on the web. That's particularly true when it comes to issues like scripting languages or navigating the Unix file system, none of which are specific to systems administration, though Nemeth describes them in considerable detail. Even the system administration aspects are mainly limited to a few specific non-BSD systems. Even if I use one of the systems they describe, the content for the other systems is of little use to me, and in all probability, the information that does interest me is out of date as this kind of book tends to have about one edition every five years, far less frequent than software updates.

The authors of Limoncelli have chosen a completely different set of skills to describe. They paint the big picture: how to survive as a system

administrator in a big, or at least medium-sized, company. Their idea of a small company is one with 100 employees. There are mentions of specific software here and there, but the book is equally applicable to BSD, Solaris, Linux, or even Microsoft. They don't reinvent the wheel. What they did put together is big enough without these details, over 1,000 pages, and more of the book would be relevant to the average reader.

One minor criticism: I think the title is wrong. This is a book about (large) systems management, not administration. And for me the penny dropped on page 364, which discusses the rationale of the customer care interface. The core of this book is about personal interactions, not nitty-gritty technical details. By contrast, Nemeth puts this issue as a small sideline at the end of the book.

One of the big issues with a book like this is where to start. No two readers want to know the same thing. And it takes a while to read through 1,000 pages of text. The authors address this problem by making chapter 1 a kind of annotated index: What to do when..., including such topics as documenting policies, surviving major outages, finding a new job, and optimizing dish washer output (presumably a joke, but I still don't understand it). In all, there are 48 different ways of looking at the book. I can see that the authors wished they could hyperlink dead trees.

The rest of the book is divided into four sections. Each chapter has two halves, titled The

Basics and The Icing, followed by a short summary (Conclusion) and a series of exercises. The exercises are to think about rather than to reply correctly. For example:

How many email namespaces are there at your site? Who controls them?

There are many small case studies, most less than half a page long, illustrating real-world cases related to the text. Frequently the case studies relate to only one of the authors, and in some cases they show differences of opinion between the authors. I find this excellent: it gives you a much better idea of what is undisputed best practice, and what is subject to personal preference.

The first section, Foundation Elements, deals with the structural aspects of system administration: computers, services, customer interaction, and much more. We're talking structure, not details. One workstation? Two? More like hundreds. And they pose completely different problems from one or two. How do you keep them up to date? How do you ensure that a nontechnical new hire has a machine that he or she can use without calling you all the time? How do you keep them running with the least effort? How do you ensure that the servers are up all the time? These are things about which I have ranted in the past, but they're all addressed in this section. There's a chapter on data centers, including practical advice on physical layout and maintaining an overview of the boxes. This seems to be about the closest the authors come to details. This section also describes issues like disaster recovery (what, no disasters? Of course there are disasters), security policy, and two chapters on customer relations, entitled Helpdesks and Customer Care. There's also a chapter about ethics.

Customer Care is a good example of the approach of the book. I'm continually ranting about help desks. Far too often you get connected to somebody who doesn't know much beyond reading a script and confirming that you have done what it says, necessary or not. The book specifically warns against too mechanical scripts that end up just frustrating the customer. Another example: escalation seems to be a non-word for many helpdesks, but it is discussed from both sides in the book. The Customer Care chapter presents and elaborates on a nine-step process for solving problems, including many options for iteration.

The second section, Change Processes, is a bit of a mixed bag. The most obviously relevant chapters are Change Management and Server Upgrades. The former covers planning and the second the execution. The topic of machine upgrades is close to my heart, and few books dare to write about it. But then there are other topics like Centralization and Decentralization and

Debugging. Particularly the latter (the first chapter in the section) doesn't really seem to fit here. As the authors say, it's related to the preceding chapter, Customer Care, in the previous section.

The next section, Providing Services, is the closest the book comes to traditional systems administration. Here you'll find advice about email, printing, and backups. But even here the emphasis is different from other books. The first chapter is Service Monitoring, with the message that if you can't monitor it, you can't administer it. Email? Nemeth discusses how to set up **sendmail**, **exim**, **postfix**. Limoncelli barely mentions individual software and is more concerned with reliability, maintainability, and scalability. What policies do you have for email addresses? What happens if somebody joins the company and has the same name as somebody already there? How do you merge email addresses when two companies merge?

The chapter on web services is in a similar vein. Strangely, though, a number of services are not covered: what about DNS? It is barely mentioned.

As the name suggests, Management Practices is the least technical of the sections of the book. The content is good, but much of it applies equally well to other disciplines. A Guide for Technical Managers caught my eye: I was in that position decades ago, and it was difficult to straddle the line between being a techie and being a manager. The contents are good, but at the end I was left feeling somewhat underwhelmed. I can't put my finger on it, and maybe it reflects more on me than the book. Although they said all the right things, I found it difficult to relate to it. Your mileage may vary.

There are two appendices: one describing the acronyms, and 26 pages entitled The Many Roles of a System Administrator. They're not all positive! You won't find the BOFH here, but there are plenty of things that administrators can do wrong, and this helps you identify them.

So what's wrong with the book? Very little. Much of the methodology seems bureaucratic, but good procedures are essential in their environment. It feels professional, not geeky, and I suspect that for this reason Nemeth will retain its cult status. But when I need to consult a book on systems administration, I'll choose Limoncelli. ●

GREG LEHEY is a retired kernel hacker. He is a committer and ex core team member of the FreeBSD Project, and was a committer to the NetBSD project. He is the author of the *Vinum* volume manager. His books include *Porting UNIX Software* and *The Complete FreeBSD*.

svnUPDATE

by Glen Barber

THIS SVN UPDATE COLUMN NORMALLY CONTAINS INFORMATION ON FEATURES BEING ADDED TO FREEBSD THAT WOULD APPEAR IN AN UPCOMING RELEASE. THIS ISSUE, HOWEVER, OVERLAPS WITH THE FINAL FEW WEEKS OF THE 10.2-RELEASE CYCLE.

10.2-RELEASE Is Imminent

In lieu of outlining features that are due to appear in an upcoming release, it seemed a perfect opportunity, on behalf of the FreeBSD Release Engineering Team, to thank the FreeBSD community and FreeBSD developers for all of the hard work that went into the upcoming FreeBSD 10.2-RELEASE.

Please be sure to browse through the FreeBSD 10.2-RELEASE notes, available at <https://www.freebsd.org/releases/10.2R/relnotes.html> for what is to be expected in this exciting new release. At the time of this writing, the release is still in progress—and hopefully will

already be available by the time this *FreeBSD Journal* issue has been published.

Thank you for supporting the FreeBSD community, *FreeBSD Journal*, and, of course, the FreeBSD Foundation. ●

As a hobbyist, **Glen Barber** became heavily involved with the FreeBSD project around 2007. Since then, he has been involved with various functions, and his latest roles have allowed him to focus on systems administration and release engineering in the Project. Glen lives in Pennsylvania, USA.



EuroBSDCon
Stockholm, Sweden
October 1–4



EuroBSDcon is the European technical conference for users and developers of BSD-based systems. The conference will take place in Stockholm, Sweden.

Tutorials will be held on Thursday and Friday in the main conference hotel (Ibis Hotel Järva), while the shorter talks and papers program is on Saturday and Sunday at the University of Stockholm.

Keynote Speaker: Paul Vixie

Founder/CEO of Farsight Security, Inc

eurobsdcon.org

Become a Sponsor, contact us at
sponsors@eurobsdcon.org

THE INTERNET NEEDS YOU

GET CERTIFIED AND GET IN THERE!

Go to the next level with



BSD
CERTIFICATION

Getting the most out of
BSD operating systems requires a
serious level of knowledge
and expertise ● ● ● ● ● ● ● ●

SHOW YOUR STUFF!

Your commitment and
dedication to achieving the
BSD ASSOCIATE CERTIFICATION
can bring you to the
attention of companies
that need your skills.

NEED AN EDGE?

● **BSD Certification can
make all the difference.**

Today's Internet is complex.
Companies need individuals with
proven skills to work on some of
the most advanced systems on
the Net. With BSD Certification

**YOU'LL HAVE
WHAT IT TAKES!**

BSDCERTIFICATION.ORG

Providing psychometrically valid, globally affordable exams in BSD Systems Administration



this month

In FreeBSD

BY DRU LAVIGNE

On July 24–26, I had the opportunity to participate in a FreeBSD Hackathon held at the Linuxhotel in Essen, Germany. This event was an experiment in bringing together a group of FreeBSD committers for a weekend of hacking in a relaxed setting. Afterwards, I interviewed **LARS ENGELS** (lme@), one of the Hackathon's organizers, to get his thoughts about the event.



Q Tell us a bit about yourself. How did you get started with FreeBSD and what is your involvement in the FreeBSD Project?

A After struggling with several Linux distributions, I discovered FreeBSD 4.7 or 4.8. With the help of the fantastic FreeBSD Handbook and the German BSD mailing list, I installed FreeBSD and learned a lot about it. A short time later, I broke the Windows installation that resided on the same disk and was too lazy to fix it, so I stayed with FreeBSD, which met most of my needs for a desktop. Since then, I use it most of the time on my notebooks and desktops as well as on my servers. When some of the programs I was using on Windows (mostly Java programs) were not part of the FreeBSD ports tree, I began to dive into the ports system and created my first ports. Having too much free time, I then started to look for unmaintained ports that have new versions upstream, updated them, and took maintainership for them. In 2007, Martin Wilke (miwi@) asked if I would like to become a ports committer and I happily accepted.

Additionally, I am on the FreeBSD forums administration team, have made some trivial fixes to this and that, translated

a short chapter of the Handbook into German, and you can often find me at the FreeBSD booth at events like FOSDEM, FrOSCon, and other smaller events.

Q DevSummits have been held during BSD conferences for several years now. What prompted you to organize a smaller, regional event? What were your goals and do you think you were able to achieve those goals?

A I knew that the Linuxhotel offers to host community events at very reasonable prices and thought it was a very good venue for a DevSummit. When I talked with Benedict Reuschling (bcr@) about it, he offered to help me organize the event and suggested running a Hackathon instead of a regular DevSummit. We both liked the idea of an event that is focused on working on code or documentation instead of presentations and talks. While the latter is important, there are already several traditional DevSummits during BSD conferences, while FreeBSD Hackathons are rare.

We did not have a specific topic or a common goal. Participants set their own goals for the weekend and, as far as I know, most of these were met.

Q For readers who are interested in organizing a Hackathon in their regions, what steps are required to organize the event and how much time is involved?

A It is actually not as much work as I thought. When the idea came to me, I found a useful guide for running a DevSummit on the FreeBSD wiki. The first “golden rules” listed there were very true: find an appropriate venue, start organizing roughly around six months before the event, and be persistent when it comes to invites and reminders. Sending emails to the FreeBSD developers mailing list and contacting local BSD and Linux usergroups is a good start for finding interested people. Be sure to send reminders so undecided people have another chance to consider if they would like to attend.

When you know how many people will come, organize food and tables at restaurants. Also, think of some social events for the evenings.

Q What types of people attended the Hackathon and what schedule developed during the event?

A We had both locals and attendees from the USA, Canada, Sweden, Belgium, and other parts of Germany. All attendees were current or former FreeBSD committers. Their area of work included all parts of FreeBSD: docs, src, and ports, as well as a developer of the OpenBSM project.

On the first day, people arrived between late afternoon until late evening and we met in

the park, enjoyed a barbecue, and worked on some docs, ports, and src bugs.

Saturday was a stormy and rainy day so we spent it inside the hotel. Sean Chittenden gave an interesting introduction to DTrace and Brendan Gregg's flame graphs, which he uses at work to find bottlenecks in the company's software. Kristof Provost explained how he flashed FreeBSD on his TP-Link router and how he configured his rather complex wireless setup. The hotel's wireless was provided by the very same TP-Link routers, but we resisted the temptation to replace Linux with FreeBSD on them. Later, we went to a nearby restaurant and had some tasty schnitzels. We spent the rest of the day in the hotel's fireplace room hacking on stuff.

Sunday was a warm and sunny day and we spent all day outside in the park, working on this and that and having another barbecue for lunch. In the early evening, everybody left and got a "FreeBSD Hackathon 2015" bag with some sweets inside as a present.

During the weekend we made 50 commits to the doc, src, and ports repositories as well as 4 to the OpenBSM GitHub repository and moving the BSMtrace repository from Perforce to GitHub.

Q Tell us a bit about the Linuxhotel and how you became aware of this venue.

A The Linuxhotel is a villa with a surrounding park that is used as a training facility for open-source operating systems, includ-

ing Linux, FreeBSD, OpenBSD, software like Tomcat and MySQL, and language courses like Perl, C, and Java. Some years ago, my brother gave the Apache course there and that is what made me aware of it. Eventually, I asked the Linuxhotel's owner if he would like me to give an introductory course to FreeBSD and he liked the idea. So now the course is offered biannually. In addition to that, Benedict and I are planning to develop an OpenZFS course.

Q Do you think you will organize another event next year? Having gone through the process, is there anything you would do differently?

A This year's Hackathon was a very nice event and hopefully the attendees had a lot of fun and would like another one next year. This was the first time we organized such an event, so not everything went as smooth as we would have liked. But, according to a survey we ran after the Hackathon, people were mostly satisfied with it.

Next time we should try to attract more non-developers and maybe people from the other BSDs. We could also encourage people to give short talks on things they work on and then collaboratively hack on them. Another idea is to have a topic of the day or run the whole event specific to a single topic. •

Dru Lavigne is a Director of the FreeBSD Foundation and Chair of the BSD Certification Group.

HERE ARE THE LINKS FOR THE RESOURCES MENTIONED IN THIS ARTICLE.

<https://wiki.freebsd.org/201507DevSummit>
<http://www.linuxhotel.de/>
<https://fosdem.org/2016/>
<http://www.froscon.de>
<https://wiki.freebsd.org/DevSummitHowTo>
<http://www.trustedbsd.org/openbsm.html>
<http://www.brendangregg.com/flamegraphs.html>
<http://www.trustedbsd.org/bsmtrace.html>
<http://www.linuxhotel.de/kurs/freebsd/>

WE Can't Do This Without YOU! YOUR CONTRIBUTION MAKES A REAL DIFFERENCE!

HELP THE FREEBSD
FOUNDATION SUPPORT:

- Project Development
- FreeBSD Advocacy
- Growth of the FreeBSD Journal
- And More!

The
FreeBSD
FOUNDATION



PLEASE DO YOUR PART
& DONATE TODAY @

freebsdfoundation.org



Events Calendar

The following BSD-related conferences will take place in the next 2 months. More information about these events, as well as local user group meetings, can be found at www.bsdevents.org.



vBSDCon • Sept 11–13 • Reston, VA

<http://vBSDcon.com> • The second biennial vBSDCon conference will take place at the Sheraton in Reston. This event begins with a one-day FreeBSD Developers Summit followed by two days of presentations and BOF sessions. The BSDA certification exam and BSDP lab beta exam are also available during this event. Registration is required for this event.

womENCourage • Sept 24–26 • Upsalla, Sweden

<http://womencourage.acm.org/> • The aim of this conference is networking and exploring career opportunities for women in computer science and related disciplines. Both men and women are welcome at this event. Several members of the FreeBSD Project will be participating on a panel about the benefits of open source and one will be giving a workshop on getting started with open source. There will also be a FreeBSD booth in the career fair. Registration is required for this event.



Open Help • Sept 25–28 • Cincinnati, OH

<https://conf.openhelp.cc/> • Open Help brings together leaders in open-source documentation and support, as well as people from across the technical communications industry who are interested in community-based help. Several FreeBSD doc committers will be in attendance and there will be a doc sprint during the conference. Registration is required for this event.



EuroBSDCon • Oct 1–4 • Stockholm, Sweden

<https://2015eurobsdcon.org/> • The primary European BSD conference will take place this year at Stockholm University. There will be a two-day Developer Summit, two days of tutorials, two days of presentations, and the opportunity to take the BSDA certification exam. Registration is required for this event. Paul Vixie will deliver this year's keynote.



OhioLinuxFest • Oct 2 & 3 • Columbus, OH

<https://ohiolinux.org/> • This is the 13th annual Ohio LinuxFest. There will be a FreeBSD booth in the expo area and several FreeBSD-related presentations. Registration is required for this event, but is free.



BSDCon Brasil • Oct 9 & 10 • Fortaleza, Brazil

<http://www.bsdcon.com.br/> • This conference is organized by the Brazilian FreeBSD users group and will feature presentations for users, enthusiasts, and developers of FreeBSD, OpenBSD, NetBSD, DragonflyBSD, and other BSD-derived operating systems. The BSDA exam will also be available at this event (in English). Registration is required for this event and there are discounts for students and professors.



Grace Hopper • Oct 14–16 • Houston, TX

<http://gracehopper.org/> • The Grace Hopper Celebration of Women in Computing is the world's largest gathering of women technologists. There will be a FreeBSD Foundation booth in the expo area. Registration is required for this event.



All Things Open • Oct 19 & 20 • Raleigh, NC

<http://allthingsopen.org/> • This annual conference explores open source, open tech, and open web in the enterprise. There will be a FreeBSD booth in the expo area. Registration is required for this event.

OpenZFS DevSummit • Oct 19 & 20 • San Francisco, CA



OpenZFS

http://open-zfs.org/wiki/OpenZFS_Developer_Summit_2015 • The third annual OpenZFS Developer Summit will be held in San Francisco and will coincide with the 10th anniversary of the open sourcing of ZFS. All OpenZFS developers are invited to participate. Registration is required for this event.

Are you aware of a conference, event, or happening that might be of interest to FreeBSD Journal readers? Submit calendar entries to editor@freebsdjournal.com.

SUBSCRIBE TODAY



“It's Awesome! This publication is the best way to popularize FreeBSD!!” — San Jose, CA

“I've found it so practical, and great reading...it caters to all levels.” — Brooklyn, NY

“The content is SO GOOD! Everyone involved in BSD should read **FreeBSD Journal!**” — Austin, TX

FreeBSDTM JOURNAL
www.freebsdjournal.org

**AVAILABLE AT
YOUR FAVORITE
APP STORE NOW**



1 YEAR \$19.99 • SINGLE COPIES \$6.99 EACH